Improving the training of Physics Informed Neural Networks via Collocation Point Sampling

Jose Ignacio Lopez Florido

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

The University of Leeds

EPSRC Centre for Doctoral Training in Fluid Dynamics

March 2026

Intellectual Property Statement

I, Jose Ignacio Lopez Florido, confirm that the work submitted is my own and that appropriate credit has been given where reference has been made to the work of others.

Chapter 4 contains work from the following publication: Florido, J., Wang, H., Khan, A., Jimack, P.K. (2024). Investigating Guiding Information for Adaptive Collocation Point Sampling in PINNs. In: *Computational Science – ICCS 2024. ICCS 2024. Lecture Notes in Computer Science, vol 14834. Springer, Cham*, DOI: 10.1007/978-3-031-63759-9_36 Florido et al. [2024]. The candidate was responsible for the research work, prepared the original manuscript and revised the manuscript. All co-authors are thesis supervisors and contributed to conceptualisation, supervision, and review of the manuscript.

This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

The right of Jose Ignacio Lopez Florido to be identified as Author of this work has been asserted by Jose Ignacio Lopez Florido in accordance with the Copyright, Designs and Patents Act 1988.

© 2026 The University of Leeds and Jose Ignacio Lopez Florido.

Acknowledgements

First and foremost I would like to thank my supervisors Peter, Amir and He for guidance and wisdom throughout my PhD. I would also like to extend my thanks to all the staff at the Centre for Doctoral Training in Fluid Dynamics as well as all the academic staff from the University of Leeds that have taught and inspired me throughout my degrees.

I would also like to thank all of my fellow PhD colleagues at the CDT for their aid and friendship; I particularly want to thank Mostafa and Usamah for their daily encouragement; and Rose, Joe, and Emily for their support and advice. My gratitude is also due to all my close friends from my time in Leeds and from back home for their friendship over the years.

Finally, I would like to thank my non-English speaking family in Spanish, for supporting me from afar throughout these years. Me gustaría tambien agradecer el apoyo de mi familia: gracias a mi prima Natalia; mis Tías Raquel, Ana, Loli y Virginia; y a mi Tio Pérez, por su compañía y apoyo mientras terminaba mi tésis. También a mi Tía Maribel por siempre apoyar mi educación, Neida por ser un gran modelo a seguir, y sobre todo, a mi madre Inmaculada por su amor y apoyo incondicional.

Abstract

Physics-Informed Neural Networks are a subset of machine learning algorithms that are trained by minimising equation residuals, as well as any available data or boundary information. They incorporate physics knowledge by measuring these residuals at collocation points sampled throughout the problem domain. Whilst the cost and accuracy of training is dependent on the number and distribution of these points, optimised sampling procedures have not been widely developed and adopted, with many applications using uniform random distributions or simple residual-driven adaptive sampling.

In this thesis, alternatives to the local residual are implemented as sources of information to guide adaptive resampling of collocation points. The performance of different guiding indicators is evaluated on 1D time-dependent PDEs, where it is shown that adaptive resampling can reduce the number of collocation points required to reach a given accuracy. Using gradients of the local residual and curvature of solution variables, in particular, is shown to provide greater benefit to accuracy and cost effectiveness.

The methodology is then extended to a system of time-dependent PDEs in two space dimensions, making necessary adjustments to architecture and training strategy. In this setting, similar improvements to performance are obtained through adaptive resampling, but alternative indicators are found to be less impactful and overall are unable to achieve error levels comparable to the 1D case. Partitioning methods are implemented to improve the robustness of training, but these are not found to circumvent scaling costs, and the use of a candidate set of collocation points, an underlying step in many adaptive sampling procedures and a contributor to this poor scalability, is identified as an important area for improvement in future work.

Contents

1 Introduction **1**

1.1 Motivations 1

1.2 Fluid Dynamics and Partial Differential Equations 3

1.3 Deep Learning in Scientific Computing 5

1.4 Physics Informed Neural Networks 7

1.5 Thesis Overview 9

2 Literature Review **11**

2.1 Machine Learning for solving Partial Differential Equations 12

2.1.1 Brief History of Scientific Machine Learning 12

2.1.2 Physics-Informed approaches 13

2.1.3 Software 15

2.2 Physics Informed Neural Networks 16

2.2.1 Architectures 17

2.2.2 Other structural developments 17

2.2.3 Understanding training and failure 18

2.2.4 Training developments 18

2.2.5 Adaptivity in PINNs 20

2.3 Adaptivity and Collocation Point Sampling 21

2.3.1 Point weighing 22

2.3.2 Initial Point Sampling 23

2.3.3 Point Resampling 24

3 Sampling the Landscape **29**

3.1 Introduction 29

3.2 Methods 30

3.2.1 Benchmark Problem: Burgers' Equation 30

3.2.2 Generating Ground Truth Solutions via FDM 31

3.2.3 Vanilla PINNs model and manual point biasing 32

3.2.4 Validation of residual based adaptive sampling methods . . . 33

3.3 Results 36

3.3.1 Manual point biasing 36

3.3.2 Validation . . . 37
3.4 Discussion . . . 38

4 Guiding Information in 1D **40**
4.1 Introduction . . . 40
4.2 Methods . . . 41
4.2.1 Benchmark problems . . . 41
4.2.2 Construction and validation of ground truth solutions . . . 44
4.2.3 Implementation of guiding indicators . . . 45
4.2.4 Hyperparameter Tests . . . 48
4.2.5 Number of repeats . . . 50
4.3 Results . . . 52
4.3.1 Burgers' Equation Results . . . 52
4.3.2 Viscosity parameter, ν . . . 56
4.3.3 Allen Cahn Results . . . 58
4.4 Discussion . . . 60
4.4.1 Overview . . . 60
4.4.2 Limitations and Next Steps . . . 61

5 Generalisability to Higher Dimensions **64**
5.1 Introduction . . . 64
5.2 Problem Description and Methodology . . . 65
5.2.1 2D Burgers' System of Equations . . . 65
5.2.2 Ground truth generation and validation . . . 66
5.2.3 Resampling in 2D . . . 66
5.2.4 Memory Batching . . . 67
5.3 Results . . . 69
5.3.1 Original Approach . . . 70
5.3.2 Resample frequency . . . 75
5.3.3 Modified Approach . . . 76
5.3.4 Higher ν case . . . 80
5.4 Discussion . . . 84
5.4.1 Overview and General Remarks . . . 84
5.4.2 Limitations and Next Steps . . . 86

6 Improvements to Higher-Dimensional case **88**
6.1 Introduction . . . 88

6.2 Methods . . . 90
6.2.1 Fixed and Adaptive set partitioning . . . 90
6.2.2 PDE based partitioning . . . 91
6.2.3 Fixed and PDE-based set partitioning . . . 92
6.2.4 Resampling "Dropout" . . . 92
6.3 Results . . . 93
6.3.1 Fixed and Adaptive subsets . . . 94
6.3.2 PDE subsets . . . 98
6.3.3 Merged subset approach . . . 100
6.3.4 Dropout or Reservoir Sampling . . . 102
6.4 Discussion . . . 103
6.4.1 Discussion of Results . . . 103
6.4.2 General Remarks . . . 105

7 Conclusion **108**
7.1 Summary of findings . . . 109
7.1.1 Chapter 4 - Aims and accomplishments . . . 109
7.1.2 Chapter 5 - Aims and accomplishments . . . 110
7.1.3 Chapter 6 - Aims and accomplishments . . . 111
7.2 Concluding remarks . . . 112
7.2.1 Assessment of overall aims . . . 112
7.2.2 Contributions to literature, limitations and future work. . . . 113

A Additional Figures **131**
A.1 Visualisation of collocation point distribution . . . 131

B Poisson Equation **135**
B.1 Introduction . . . 135
B.2 Method . . . 136
B.3 Results . . . 138
B.4 Discussion . . . 141

List of Figures

3.1 Reference solution of 1D Burgers' Equation with $\nu = \frac{0.01}{\pi}$ 31
3.2 Grid independence study of FDM solution. 32
3.3 Illustration of manual bias of collocation points 34
3.4 Plot of L_2 error against bias 37

4.1 Reference solution of Burgers' Equation for various values of ν . . . 42
4.2 Example of initial condition generation 44
4.3 Reference solutions to Burgers' Equation with alternative initial conditions. 45
4.4 Reference solution to the Allen-Cahn equation. 46
4.5 Grid independence study for alternative initial conditions 46
4.6 Hyper-parameter investigation across different guiding indicators. . . 50
4.7 Hyperparameter investigation for residual-derivative indicator 50
4.8 Convergence plot of cumulative average error against number of repeats 51
4.9 Plot of error against resamples using adaptive and non-adaptive methods 53
4.10 Plot of error against number of collocation points using adaptive and non-adaptive methods 54
4.11 Plot of time against number of resamples using adaptive methods with different guiding information 55
4.12 Plot of error against number of resamples for alternative initial conditions 56
4.13 Plot of error against resamples for viscosity $\nu = 0.01$ 57
4.14 Plot of error against resamples for viscosity $\nu = 0.001$ 57
4.15 Plot of error against resamples, solving Allen-Cahn with $N = 500$ . . 59
4.16 Plot of error against resamples, solving Allen-Cahn with $N = 1{,}000$. 59
4.17 Plot of error against resamples, solving Allen-Cahn with $N = 2{,}000$. 60

5.1 Reference solution of a 2D Burgers' system of equation 66
5.2 Error against size of candidate point set 68

5.3 Plot of error against number of collocation points using adaptive and non-adaptive methods, with network depth 3 and 50 resamples . . . 70
5.4 Plot of loss history with various N, using a non-adaptive method with network depth of 3 . . . 71
5.5 Plot of error against number of collocation points using non-adaptive methods and varying network depth . . . 72
5.6 Plot of loss history using residual resampling with network depth of 3 and 50 resamples . . . 73
5.7 Ground truth, PINN prediction and error contours of u using 50-resample approach . . . 74
5.8 Ground truth, PINN prediction and error contours of v using 50-resample approach . . . 74
5.9 Plot of error against number of collocation points using various configurations of number of resamples . . . 76
5.10 Plot of error against number of collocation points, using a modified 5-resample approach . . . 77
5.11 Plot of cost against number of collocation points using modified approach . . . 77
5.12 Plot of loss history using residual resampling, N=50,000 and a network depth of 5 . . . 78
5.13 Ground truth, PINN prediction, and error contours of u using modified approach . . . 79
5.14 Ground truth, PINN prediction, and error contours of v using modified approach . . . 80
5.15 Plot of error against number of collocation points using modified approach, $\nu = 0.01$ . . . 80
5.16 Ground truth, PINN prediction and error contours of u, $\nu = 0.01$ . . . 81
5.17 Ground truth, PINN prediction and error contours of v, $\nu = 0.01$ . . . 82
5.18 Plot of loss history using residual resampling and the modified approach, $\nu = 0.01$ . . . 82
5.19 Plot of loss history using non-adaptive methods for various N, $\nu = 0.01$ 83
5.20 Plot of cost against number of collocation points, $\nu = 0.01$ . . . 84

6.1 Plot of error against N using fixed-adaptive partition, $c = 0$ . . . 94
6.2 Plot of error against N using fixed-adaptive partition, for various partition fractions, $c = 1$ . . . 95

6.3 Plot of error against partition fraction for various N, $c = 1$ 95
6.4 Plot of error against N for various partition fractions, $c = 0$ 96
6.5 Plot of error against partition fraction for various N, $c = 0$ 96
6.6 Plot of error against N for three fixed-adaptive partitioning methods 97
6.7 Plot of error against N using PDE partition 99
6.8 Plot of error against $N$for various partition fractions 99
6.9 Plot of error against partition fraction, for various N 99
6.10 Plot of error against N, using three-way partitioning 101
6.11 Plot of error against N using different partitioning strategies 102
6.12 Plot of error against fraction of points ignoring resampling 102

A.1 Point Distributions Histogram for Burgers' 1D Equation 132
A.2 Point Distributions Histogram for Burgers' 2D Equation 134

B.1 Poisson reference solution mesh independence 136
B.2 Approximation Distance Function derived transformation 137
B.3 Plot of error against N, solving Poisson with adaptive and non-adaptive methods 138
B.4 Plot of error against N, solving Poisson with different adaptive re-sampling training strategies 139
B.5 Ground truth, PINN prediction, and error contours, of fixed versus adaptive methods 139
B.6 Ground truth, PINN prediction, and error contours with different adaptive resampling training strategies 140
B.7 Point Distributions Histogram for Poisson 2D Equation 140
B.8 Plot of loss history using fixed distribution methods 141
B.9 Plot of loss history using adaptive resampling methods 142

List of Tables

3.1 Comparison of L_2 error obtained using RAD and RAR-D to literature 38

4.1 Table listing notation for different guiding indicators 48
4.2 L_2 Error results of different adaptive and non-adaptive methods, $\nu = 0.01$, $L = 100$ 57
4.3 L_2 Error results of different adaptive and non-adaptive methods, $\nu = 0.001$, $L = 100$ 58

5.1 Table showing configurations of number of resamples and optimisation steps per resample 75

Chapter 1

Introduction

1.1 Motivations

The study of fluid dynamics plays a role across many scientific and engineering disciplines, and provides insights into the behaviour of fluids in a wide range of scenarios and applications, from meteorology to biological and supersonic flows. Traditionally, the solution of fluid dynamics problems has relied on numerical methods ranging from Finite Element (FE) methods to advanced Computational Fluid Dynamics (CFD) simulations via Finite Volume (FV) software, such as open-source solvers like OpenFOAM Jasak et al. [2007] or commercial packages. While our understanding today of fluid flow phenomena is underpinned by research using these numerical methods, many have also looked at the recent successes of data-driven methods in fields of computer vision and language processing and considered how to adapt them for tackling these problems too.

In this subsection, we will explore how Physics Informed Neural Networks (PINNs) have emerged from the field of Machine Learning (ML) as an alternative deep learning method with unique advantages for solving scientific computing problems. We will thus motivate the research carried out in this thesis into collocation points, the defining feature of PINNs (described in technical detail in section 1.4).

With advancements in hardware capabilities and increasing availability of High Performance Computing (HPC), even expensive computational simulations such as Large Eddy Simulations (LES) and Direct Numerical Simulations (DNS), which fully resolve increasingly smaller scales, have become feasible as research tools. However, the computational costs associated with these approaches remain prohibitively expensive for most practical applications and design processes [Molland and Turnock, 2007]. Although the development of HPC technology to further increase hardware capabilities is an ongoing research topic of its own [Al-Shafei et al., 2022], for these

methods, increasing accuracy requires either refinement of the mesh or increasing the polynomial degree (in the case of FE), both of which increase costs nonlinearly as problem dimensionality increases. Since this scaling is intrinsic to these methods, a gap exists for methods with different underlying frameworks with better cost scaling properties.

At the same time, in recent years there has been a remarkable rise in the utilisation of machine learning algorithms for a variety of applications. Deep neural networks in particular have demonstrated capabilities in pattern recognition, data analysis, and decision-making tasks. The ability to learn from data and extract meaningful patterns has led to significant advancements in image and speech recognition, natural language processing, and many other fields, some of which are discussed in section 2.1. These characteristics highlighted the potential of ML and spurred efforts to merge ML approaches with CFD (Section 2.1). Once trained, machine learning models can generate solutions very quickly compared to numerical methods. However, the accuracy of these models is dependant on the resolution of data available and even in data-rich scenarios they may struggle to produce results of comparable accuracy. Often this requires having solution data obtained from numerical models to begin with, and the ability of the models to generalise to unseen problems is an important issue as the risk is accuracy is high only within the trained dataset. In 2023, Cueto and Chinesta [2023] argued "neural networks are not very popular among the scientific community in general" due to other well known problems with ML such as overfitting, the impact of perturbations in data, or the already mentioned lack of generalisation and corresponding reliance on traditional methods data to train on. However, whilst ML methods alone may thus not surpass classical solvers in accuracy, their strengths lie in offering potentially favourable trade-offs between accuracy and speed. Such trade-offs may be beneficial in applications such as real-time control, where fast resolution times are critical; or design-space exploration, where a large number of comparatively coarse evaluations are required.

The combination of the above factors has driven research into what potential uses ML may have in engineering and science applications broadly. Within this background, PINNs (Section 1.4) have emerged as a specific type of neural network (NN) architecture that attempts to bridge some of these ML challenges by harnessing information not usually present in ML: knowledge of the physics. In an example scenario where prior knowledge informs us of symmetry of the solution; this property could be enforced to obtain a physical solution requiring data to resolve only a frac-

tion of the domain. PINNs extend this concept, utilising knowledge of the governing Partial Differential Equation (PDE) of a problem as a soft constraint in the optimisation process of the network. This reduces the dependence on obtaining a dataset via other means, allowing PINNs to function with much reduced datasets, or even to function in the absence of data. Thus, in addition to the previously discussed benefits of ML, we can also see how PINNs offer a much greater flexibility, being able to deal with sparse or noisy data, unknown boundary conditions and complex geometries, thanks to the robustness offered by its multiple sources for optimisation. These features suggest the potential usefulness of PINNs for scenarios where moderate accuracy is acceptable in exchange for greater flexibility or reduced data requirements, with some examples in the fields of fluid dynamics such as inverse problem solving or field flow reconstruction from sparse data Cuomo et al. [2022] Farea et al. [2024].

As a relatively new area of research, much of the work carried out on PINNs has been exploratory in various directions and there remain many gaps. As will be discussed in chapter 2, an essential component of PINNs, namely the collocation points used to implement physics constraints, are frequently selected in a naïve manner. For this reason, in order to contribute to further developing the effectiveness of PINNs for scientific applications such as fluid dynamics research, this thesis is concerned with the selection of the PINNs' collocation points for the solution of selected relevant PDEs, and the effect of this selection on model error.

1.2 Fluid Dynamics and Partial Differential Equations

Partial Differential Equations (PDEs) are equations or systems of equations involving unknown functions of several variables and their partial derivatives. For a simple example of this we present the wave equation, where the displacement field $u(x,t)$ depends on both space x and time t:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \tag{1.1}$$

where c is a coefficient representing the speed of the wave.

In two space dimensions and higher, second derivatives with respect to the spatial coordinates are described by the Laplacian Operator Δ. The same wave equation for $u = u(x,y,t)$ in Cartesian coordinates (x,y) then becomes:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \, \Delta u, \tag{1.2}$$

where the Laplacian is defined as

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}. \tag{1.3}$$

In equation 1.1 the dependent variable u is a scalar quantity; however, in a fluid dynamics context, the functions we solve for tend to be velocity vector fields depending on both time and space.

In fluid dynamics, problems are usually governed by two equations: conservation of mass and conservation of momentum. Conservation of mass is often enforced via a continuity equation, ensuring the mass of fluid entering and leaving each region of the domain is equal:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{1.4}$$

where ρ is the fluid density and $\mathbf{u}$ the fluid velocity.

Meanwhile, conservation of momentum in Newtonian fluids is described by the Navier-Stokes equations, which account for additional terms like pressure, viscous stress, and external body forces:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{1.5}$$

where p is pressure, μ is dynamic viscosity, and $\mathbf{f}$ represents external body forces. This system governs how momentum is transported and diffused through the fluid, the left-hand side represents advection, while the right-hand side accounts for pressure, viscous effects, and body forces respectively.

In many applications, it is unnecessary to solve the full Navier–Stokes system, as assumptions or symmetries can often be exploited to arrive at simplified models of the flow. Similarly, simplified models that capture certain aspects of fluid behaviour are also used in many applications. An example of this is the one-dimensional viscous Burgers' equation, a common fluids benchmark that models advection and diffusion:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}, \tag{1.6}$$

where ν represents viscosity.

However, to physically constrain the problem and obtain a unique solutions, initial conditions (at $t = 0$) and boundary conditions (at the edges of the spatial domain) must be specified. Several types of boundary condition exist. In Dirichlet conditions, the value of a variable (e.g. velocity or pressure) at the boundary is

prescribed. In Neumann conditions, the derivative of the variable normal to the boundary is prescribed (describing a flux into or out of the domain). More complex boundary conditions, such as symmetry, periodic, or mixed types, can also be imposed, though they will not be considered explicitly in this thesis.

1.3 Deep Learning in Scientific Computing

Machine learning covers a broad set of methods, where the goal is to map inputs to outputs via learned functions. To illustrate how these work, this subsection introduces some of the core ideas of regression-based networks, where the model learns to approximate continuous functions. This is in contrast to tasks like classification (where the goal is to assign discrete categories to the inputs) or reinforcement learning (where training is driven by interactions between the model and an environment, rather than direct error minimisation). To look at regression, we will consider the example of a fully connected neural network, a sub-class of ML model, to examine its structure and components, its key characteristics, and its training process. These concepts allow us to introduce the PINNs framework in the next subsection and to better understand its novelty, as well as to understand how the adaptive sampling methods developed later in this thesis interact with the training process of the network.

The structure of a neural network consists of layers of units (frequently referred to as "neurons" by analogy with synapses in the brain) which each represent simple mathematical operations. The output of the operation travels forward, and when outputs meet a linear combination is taken. A full pass through the network from the input to the output of the final layer is known as a 'forward pass'. To allow the network to express nonlinear solutions, the outputs to each neuron are connected to 'activation functions', nonlinear functions such as $\sin(x)$, ReLU, or tanh.

To understand how the model solution can change, we now examine the form of the mathematical operation within each neuron. This operation contains weights, which scale the inputs, and a bias, which shifts the result; both are adjusted independently in each neuron during training to modify the network's response, as defined in equation (1.7).

$$z = \sum_{i=1}^{n} w_i x_i + b, \tag{1.7}$$

where x_i are the inputs, w_i are the weights, and b is the bias. These weights and biases are the parameters that can be tuned throughout the network training to

modify the response of the model.

After computing the linear combination z, the neuron then applies the selected nonlinear activation function to produce its final output z'. Using the tanh activation function as as an example (used later in this thesis), the neuron output is given by equation (1.8):

$$z' = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{1.8}$$

which maps z to a value between -1 and 1.

Having introduced the basic elements of how a neural network produces an output solution from given inputs, and how this output depends on weights and biases, we next consider how the model performance is quantified. To do this, we define a loss (or cost/objective) function. This function measures the difference between the current model prediction and the desired prediction. In 'supervised' learning problems, this difference can be calculated directly from known solution data, as the function can be defined simply as the absolute difference or mean square error, for example. In contrast, in 'unsupervised' learning solution data is not available, and so loss functions have to be defined using alternative criteria. As we will later see, PINNs use information of the governing equations in the loss function to allow the network to learn a solution even without explicit data.

The ultimate goal in the training of a network is to solve an optimisation problem that minimises the value of the chosen loss function over the possible values for the weights and biases associated with every neuron. To do this, the first step is to compute the derivatives of the loss with respect to the weights and biases in the network. This is possible thanks to a key characteristic of neural networks, which is their differentiability. All of the simple operations carried out in the network (additions, multiplications and simple activation functions) are differentiable. This allows the derivatives of the function to be computed efficiently and accurately, as the chain rule can be applied systematically in a 'backward pass' through the network, producing exact derivatives through simple arithmetic operations. For vector inputs (for example a set of input coordinates), this process generates the Jacobian matrix, containing all partial derivatives of the outputs with respect to the input vector.

Knowing the derivatives of the loss allows the optimiser algorithm to subsequently update the network parameters (the weights and biases), to improve on the model's previous solution. This process is iterative and constitutes the main computational step in gradient-based optimisation. In gradient descent the differentiability

of the network is used to find the direction in which to change each parameter to best reduce the loss function. An important hyperparameter, the learning rate, controls the size of these changes, balancing stability with speed of convergence. In full gradient descent approaches, optimisers evaluate gradients for all samples in a dataset, whilst stochastic batched gradient descent optimisers evaluate gradients at a fraction of the dataset selected at random. The latter is more computationally efficient and allows more frequent parameter updates, often leading to a faster convergence despite individual updates being less accurate.

With these fundamentals established, the next subsection will now cover the fundamentals of a physics-informed neural network. The underlying structure of a PINN is the same, so concepts explained in this section still apply, with the novelty lying in how physics is incorporated into the network's loss function.

1.4 Physics Informed Neural Networks

The distinguishing feature of PINNs is the inclusion of the 'physics' of the problem into the structure of the neural network. This is done by modifying the loss function, including additional terms representing the residuals of the governing PDE(s) ($\mathcal{L}_{PDE}$), and the boundary conditions ($\mathcal{L}_{BCs}$), to the original loss from known data ($\mathcal{L}_{data}$). These terms ensure that the network will attempt to satisfy both the PDEs and boundary conditions as part of the training process. Overall, the PINN loss ($\mathcal{L}_\theta$) can be defined as:

$$\mathcal{L}(\theta) = w_{data}\mathcal{L}(\theta)_{data} + w_{PDE}\mathcal{L}(\theta)_{PDE} + w_{BC}\mathcal{L}(\theta)_{BCs}$$

Here, the relative impact of each loss term can be controlled through the weights, $w_{data}, w_{PDE}, w_{BC}$, whilst θ represents the trainable network parameters. In the case of one-dimensional time-dependent problems for example, we will consider a spatial dimension x, a temporal dimension t and solve for the dependent variable $u(x,t)$.

The process of training a PINN is the same as for a regular NN: once an architecture and the essential parameters are defined we seek to find values of these weights and biases at each network neuron that minimise the prescribed loss function - including the $\mathcal{L}_{PDE}$ term in the loss function. This term is obtained by evaluating the PDE residual (f, say) at multiple points throughout the domain. This set of points where the residual is measured are the 'collocation points'. If we let $\mathcal{T}$ denote the full set of collocation points, and x individual points; then the PDE loss can be

described by:

$$\mathcal{L}_{PDE} = \frac{1}{|\mathcal{T}|} \sum_{\mathrm{x} \in \mathcal{T}} |f(\mathrm{x}; u; u_x, u_t, u_{xx}; \theta)|^2$$

The significance of the number and location of the collocation points is substantial, since they determine how well the PDE residual is mapped throughout the domain. Points can be placed randomly to cover the entire domain, but quasi-random methods such as Latin Hypercube Sampling (LHS) can also be employed to avoid redundant points and/or minimise distance between points.

For forward problems, PINNs can be trained without the need for data, if the boundary conditions can be enforced in a 'hard' manner. In 'hard' constraining, mappings applied to the output of the NN force the boundary conditions to hold true. This is in contrast to satisfying the boundary conditions via 'soft' constraining, which involves adding the BC loss as a term to minimise in our cost function. Throughout this thesis we adopt the former approach so as to allow us to focus exclusively on the PDE residual contribution to the loss, and therefore on the sampling of collocation points throughout the interior of the domain. Hence, all Dirichlet boundary conditions are strongly imposed by applying an output transformation to the neural network prediction, treating the PINN as an unsupervised problem with no input data. This output transformation is a manually selected mathematical operation applied to the network's output that ensures the transformed solution satisfies the required values at the boundaries. For example, for a domain $0 < x < 1$ with Dirichlet ($u = 0$) at both ends, the transformation:

$$\tilde{u}(x) = \sin(\pi x)\, \hat{u}(x)$$

ensures the transformed output $\tilde{u}(x)$ satisfies the boundary conditions regardless of the raw network output, $\hat{u}(x)$.

Another aspect of PINNs research focuses on utilising them for inverse problems, where the parameters of the PDE are unknown and solved for from data. In those cases, the overall approach is slightly different and data dependant; however since the focus of this thesis is to gain deeper understanding of the choice of the collocation points on the performance of PINNs, we do not consider such inverse problems in this work.

However, for forward unsupervised PINN problems, collocation points become the primary driver of optimisation, as the optimiser relies entirely on the information about the PDE provided at these points. For this reason, the distribution of collocation points is of great importance, as it determines both the efficiency of training and

the accuracy of the final solution. In addition to avoiding gaps in the domain and redundant clustering, evidence also suggests it may also be beneficial to place more points in regions of physical importance Wang et al. [2023b], Lu et al. [2021b]. This collocation point selection problem is the main concern of this thesis, motivated by its impact on network performance and by the potential for improvement through drawing on sampling concepts developed from other domains.

1.5 Thesis Overview

In this short chapter the necessary foundational knowledge has been covered to allow an explanation of the purpose and aims of the work. In Chapter 2, a literature review summarises the vast breadth of research relevant to this thesis. It covers the broader machine learning and physics-informed neural networks research area before focusing on adaptive strategies in PINNs and point resampling, the areas in which our thesis makes novel contributions.

Chapter 3 shows the results of preliminary investigations, which were used to guide decisions on network parameters. This chapter also introduce adaptive resampling and validates the existing methods and benchmarks. The first original contributions are then presented in Chapter 4, which investigates the use of a range of different potential indicators to guide the adaptive resampling of collocation points. The work presented in Chapter 5 extends these ideas to a higher dimension benchmark, examining how the benefits observed in the previous chapter scale. It also discusses changes needed in the original method, which is continued in Chapter 6. In Chapter 6, fundamental aspects of the resampling method itself are modified based on the insights from previous chapters. Lastly, Chapter 7 concludes the thesis by summarising the work carried out and discussing its role in the literature.

The main contributions of this thesis are thus presented across three separate chapters. However, it is important to clarify that the nature of the work is split into two distinct contributions. In Chapter 4, the main focus is on whether alternative indicators can be more effective for guiding resampling of collocation points. Essentially, this looks at the inputs to the adaptive resampling method, and how choosing certain inputs affects the quality of the resulting set of collocation points. In contrast, the work in Chapters 5 and 6 extends those methods to a higher dimensional system of PDEs and aims to improve the method by which the resampling itself is carried out.

Summary of Contributions

The specific contributions of this thesis, detailed in later chapters, are:

- Evaluation of alternative indicators for adaptive resampling (Chapter 4): A systematic investigation of multiple candidate quantities for guiding the relocation of collocation points, assessing their effectiveness relative to commonly used residual-based criteria.
- Extension and assessment of these indicators on higher-dimensional benchmarks (Chapter 5): An analysis of how the indicators identified in Chapter 4 scale to more complex domains, including the identification of limitations that motivate methodological modifications.
- Development of modifications to the resampling procedure itself (Chapter 6): Introduction and evaluation of changes to the underlying resampling mechanism, informed by insights from Chapters 4 and 5, with the aim of improving stability and solution accuracy.

Chapter 2

Literature Review

This literature review is separated into three sections. Section 2.1 covers the broad area of machine learning used for the solving of Partial Differential Equations (PDEs). It highlights noteworthy non-physics informed approaches used as alternatives to, or as aids to, traditional numerical methods. It then takes a look at how physics-informed approaches can be used to merge the advantages of both physics knowledge and machine learning methods. This will reveal the motivations behind combining available knowledge of physics with data-driven techniques, and the limitations of purely data-driven methods applied to PDEs.

Section 2.2 then focuses, within the field of physics-informed approaches, on Physics-Informed Neural Networks (PINNs). There is still a large breadth of research in this area, and we cover significant contributions to network design (architectures), training approaches, and understanding of training and failure modes. It lastly covers different kinds of 'adaptivity'. This aims to show the many areas of research involved in improving PINNs, highlighting the challenges they face in practice and what strategies, including adaptive techniques, are being used to improve accuracy and efficiency.

In section 2.3, we consider the most relevant literature to this thesis, looking at the problem of sampling of collocation points in PINNs, and how adaptivity in this context guides the refinement or replacement of collocation points throughout training. We review different methods and how they use available information from training, mainly the loss, to identify regions that collocation points should be added to, and how actually add or move these points. While there is a wide diversity of techniques, we highlight that the error metrics that are used to guide resampling almost are almost exclusively derived from PDE residuals, a limitation that the work in this thesis aims to address.

In this chapter, we aim to present a range of papers that are representative of the key research areas discussed. However, where available, review papers have also been referenced to guide readers toward more focused coverage.

2.1 Machine Learning for solving Partial Differential Equations

2.1.1 Brief History of Scientific Machine Learning

Despite the recent surge in interest in machine learning, perhaps driven by the mainstream prominence of artificial intelligence (AI) in the public consciousness, much of the research underpinning these methods has a longer history, with many of its foundational concepts being introduced several decades ago in Nilsson [1965]. However, hindered by the lack of availability of computation power at the scale we have today, the implementation and testing of these ideas involving 'universal approximators' was largely unfeasible.

As computing power increased steadily over the years, these theoretical ideas resurfaced. By the late 1980s, new research further developed the efficient back-propagation methods [Rumelhart et al., 1986] that allow tackling high input dimensions, necessary to handle image recognition in real world scenarios. This major development allowed for automated reading of handwriting in applications like the US postal service, developed in the late 80s [LeCun et al., 1989], fully developed and in use by the end of the 90s [LeCun et al., 1998]. In fact, it was work in these two decades carried out by John J. Hopfield [Hopfield, 1995], and Geoffrey E. Hinton [Rumelhart et al., 1986] that would later see them awarded the Nobel Prize in Physics in 2024, labelled as the "foundational discoveries and inventions that enable machine learning with artificial neural networks".

Additionally, work focused on utilising neural networks (NNs) for solving equations also appeared. Dissanayake and Phan-Thien [1994] proposed using these neural network concepts to solve Partial Differential Equations (PDEs) by minimising a residual loss, a concept very similar to the research presented in this thesis. A few years later, Lagaris et al. [1998] present a more general framework to also solve the initial and boundary value problems, marking another step towards practical neural-network-based PDE solvers. These early contributions laid the groundwork for later developments using increasingly complex networks to tackle scientific problems.

The next part of this brief history concerns the growth of interest in deep learning

LeCun et al. [2015], Goodfellow et al. [2017]. Through deep neural networks we see an interest in using ML to solve PDEs, extending the uses of ML into the fields of science and engineering. By this time, the growing interest in ML had led to a wide range of neural network architectures being developed for solving PDEs, developing different features to adapt to different problem settings. Some representative examples include Sirignano and Spiliopoulos [2018] solving the strong form of quasi-linear PDEs with DNNs, Zang et al. [2020] solving the weak forms of linear PDEs via adversarial networks, Han et al. [2018] high-dimensional PDEs, and Berg and Nyström [2018] solving PDEs in complex geometries.

A substantial fraction of papers published at this time deal with adapting present architectures and methods to specific fields, attempting to leverage the advantages of machine learning methods in scientific and engineering contexts. In fluid mechanics, for example, reviews such as Brunton [2021] Vinuesa and Brunton [2021] synthesise these developments, highlighting their successes and limitations. General reviews like those and opinion articles by leading researchers are useful for framing the state of the art. For instance, E [2021] discusses the role of machine learning as a tool for applied mathematics, while E et al. [2022] reviews the challenges of numerically solving high dimensional PDEs and how, amongst other numerical methods, deep learning has emerged as a promising new direction in scientific computing.

As noted by E [2021], machine learning models can be 'surprisingly powerful... [but] also quite fragile'. From the papers cited above, the potential of machine learning models to solve problems across many scientific and engineering disciplines is clear. These papers cite reasons such as the potential for multi-scale learning, fast inference times once trained, and the ability to train without full knowledge of physics and boundary conditions. However, as the comment on 'fragility' implies, careful construction of these models was required to realise these benefits.

In most engineering and science applications though, a common feature is knowledge of some, if not all, the physics involved in the problem. Consequently, an important focus for machine learning in scientific computing is on how to best integrate the known physical laws into these methods for further benefit.

2.1.2 Physics-Informed approaches

The availability of knowledge of physics for a given problem greatly extends the potential of machine learning methods. Incorporating physics can help in different ways, some examples include improving generalisation by learning universal laws, extending the possible applications by using physics to fill in gaps in datasets, or

exploiting symmetries and defining boundaries to help reduce the volume of training data required.

The implementation of physics can help act as a regulariser to solve a target regression problem from limited or noisy data, as shown in Long et al. [2019], Raissi and Karniadakis [2018], Shone et al. [2023]. When the form of an equation is known, ML can be used to learn missing parameters from data, known as an inverse problem, then using the completed equation for new problems. In the case of PINNs, implementing the physics equations into the loss function allows us to directly solve our problem by minimising our loss.

Both Han et al. [2018] and Sirignano and Spiliopoulos [2018] tackle the challenge of using machine learning to solve high-dimensional PDEs. Their success serves as proof of concept of the potential of deep learning algorithms as efficient solvers for higher dimensional problems.

Karniadakis et al. [2021] outlines the complexities and challenges of ML that incorporating physics can help with. This broad review overviews different methods of embedding physics, the merits of incorporating physics, specific examples of applications across different fields, and limitations and future outlook. A few other reviews are available that cover these topics: Sharma et al. [2023] reviews deep learning applications with a fluid mechanics perspective; Vinuesa and Brunton [2021]'s review similarly has a fluid dynamics perspective, in this case looking at how machine learning can enhance computational fluid dynamics methods; and Taebi [2022] narrows down the applications to fluid dynamics in medical environments, where specific examples are given of NNs being tested for relevant medical applications. For a researcher in these topics, reviews like the above are essential to catch up to the many distinct approaches that have been developed due to the large volume of work published every year. Having discussed those, we will next discuss some key examples of some of these physics-informed approaches.

The auto-regressive Geneva and Zabaras [2020] combines a few different approaches to solve nonlinear PDEs featuring discontinuities. It tests a dense encoder-decoder surrogate, using a physics constrained approach to alleviate the costs of generating the large amounts of training data that surrogate model would need. In addition, it includes a Bayesian framework, which is an important feature in machine learning research as it allows for uncertainty quantification helping establish confidence in machine learning results, normally a downside due to the black-box nature of many ML models.

For dynamic systems, reinforcement learning is another useful class of ML algo-

rithm where a control strategy is learned from interactions with the system. Specifically, model-based reinforcement learning (MBRL) attempts to train a model to represent the system, thus minimising the cost of using a real system for training a real flow control strategy Rabault et al. [2019]. In this type of ML, work by Liu and Wang [2021] introduces physics constraints into the framework to inform the model learning. This improves both the quality of the learned model of the system and significantly reduces the required interactions with the model to learn the control strategy, leading to better sample efficiency (requiring less real data) and learning performance, demonstrating effectiveness for dynamical systems governed by ODEs and PDEs.

One last example of how implementing physics can improve on ML methods involves the field of operator learning Lu et al. [2021a]. Here, deep operator networks learn not just continuous functions, but mappings between functions, allowing the model to learn more complex systems, for example learning multi-scale operators or multi-physics operators. When governing equations are known, these can be incorporated into the loss function to aid in cases with sparse data availability Goswami et al. [2022], where the physics loss component helps solve the underlying PDE constraints with fewer labelled data points.

The last example's physics loss component takes inspiration from Physics-Informed Neural Networks (PINNs) Raissi et al. [2019]. This specific implementation of physics-informed machine learning is the focus of this thesis, and so it will be further explained in Section 2.2 in detail. First, we will cover the underlying technical framework that enables this research: the programming languages, software and libraries.

2.1.3 Software

This subsection covers the software necessary to enable machine learning workflows. The first choice is the programming language that will be used, which forms a foundation for the subsequent software. In a research context for machine learning, the most widely used language is Python, although alternatives such as Julia have also been developed for specific AI applications Rackauckas and Nie [2017].

Using the Python programming language, different frameworks have been developed which consist of single or multiple modules that provide the necessary tools for building machine learning programs. These tools simplify the processes of coding all aspects of a machine learning problem, from selecting network architectures to carrying out training and inference. PyTorch [Paszke et al., 2019] and TensorFlow [Abadi et al., 2016] are the two most popular python frameworks, due

to their general-purpose nature. However, other frameworks do exist such as SimNet [Hennigh et al., 2020] by NVIDIA, which may attempt to offer more specialised features.

Lastly, some specific python libraries can also be of great use to researchers. A relevant example to this thesis is DeepXDE, introduced by Lu et al. [2021b] to specifically streamline the workflow of creating and testing ML models. Other libraries like JAX Bradbury et al. [2018] Frostig et al. [2019] attempt to improve computational efficiency and speed.

2.2 Physics Informed Neural Networks

A very significant development in machine learning for science and engineering was the introduction of Physics-Informed Neural Networks (PINNs), which popularise the idea of incorporating physics into the loss function through terms representing PDEs and boundary conditions. PINNs, originally introduced in Raissi et al. [2019], evolved from attempts to obtain solutions for linear differential equations in Raissi et al. [2017a,b], Owhadi [2015], and nonlinear partial differential equations (PDEs) Raissi and Karniadakis [2018], using Gaussian process regression. Whilst those attempts showed the usefulness of prior physics information, the limitations of those methods in representing nonlinear equations led to the alternative approach of using deep neural networks, which could be trained to approximate nonlinear functions in a more data-efficient manner thanks to their physics-informed design. The efficacy of PINNs to both solve and identify PDEs from data (inverse problems) using available physics information has led to a great amount of interest and PINNs becoming a research area of their own.

We distinguish two broad directions of research in PINNs: looking at the application of PINNs to specific fields or applications; and looking at developing the implementation of PINNs themselves or extensions to the original framework. For the former, the research focuses on finding effective configurations of PINNs and comparisons of their performance to existing state-of-the-art methods. Reviews by topic such as Cuomo et al. [2022] and Farea et al. [2024] cover PINNs applied to different domains, and Zhou et al. [2024], Cai et al. [2021] look at fluids applications specifically. Some of this research has also examined other types of PDEs: integro-differential [Lu et al., 2021b], fractional [Pang et al., 2019], stochastic [Zhang et al., 2019], and inverse [Lu et al., 2021c].

For the latter direction of research in PINNs, we broadly define work that at-

tempts to improve PINNs implementation more generally, as opposed to for specific purposes. In the following subsections we group papers into broad categories to show different approaches to improving PINNs. Starting with changes to NN architecture, specific additions, theoretical understanding of convergence limits and fundamental shortcomings of PINNs, improvements to the training method itself, and lastly, adaptivity.

2.2.1 Architectures

Since in PINNs, physics is implemented via the cost function, the rest of the ML algorithm design is very flexible. While the original approach in Raissi et al. [2019] is a simple fully-connected network, other structures like convolutional networks or graph neural networks can be compatible. To give some examples, Yang et al. [2021] combines a Bayesian neural network structure Geneva and Zabaras [2020] with PINNs to add uncertainty quantification. The recent Li et al. [2025] combines PINNs with a generative adversarial network (GAN) approach to improve dataset efficiency in problems with low data availability.

For example, Kolmogorov-Arnold networks Liu et al. [2025], which use learnable activation functions, have also been combined with PINNs in Rigas et al. [2024], and taken further with an encoding layer to positive effect in Zhang et al. [2025b]. These examples show the flexibility of PINNs to be combined with other different frameworks, but are not limited to the examples given.

2.2.2 Other structural developments

As well as being combined with different machine learning frameworks, PINNs are also easily compatible with other small enhancements or implementations of the known physics. In one example of this, the PINN in Anagnostopoulos et al. [2023] is enhanced by the addition of encoders from the work by Wang et al. [2021] to help improve performance. This is a similar idea to Tancik et al. [2020], which seems to improve on "coordinate-based" NNs, by mapping input coordinates to a higher space via Fourier Features, helping to learn higher-frequency solutions in computer vision problems, and in PINNs, this can help learn patterns between inputs that are farther apart in the physical domain space.

Covering something quite different, Elmaradny et al. [2025] uses the PINN formulation of minimisation but instead of considering a traditional Navier-Stokes system of equations, applies it to the principle of minimum pressure gradient (PMPG) to solve data-free fluid mechanics problems.

2.2.3 Understanding training and failure

Whilst the above developments tend to be tested empirically, a vital part of research is the more theoretical research into the limits and potential of different methods. This work on understanding is crucial for justifying the research on ML and helping identify the underlying challenges to focus on next, as well as generally helping correctly find the use cases of PINNs.

Initial studies extending the PINNs framework such as Pang et al. [2019], consider in detail the sources of error in PINNs. These detailed analyses distinguish between the sensitivity due to discretisation, point sampling, NN architecture and optimisation. These details help us understand the different aspects we have to tune, but subsequent work is needed that delves into the underlying limits and possible solutions.

Krishnapriyan et al. [2021] does this, by looking specifically at the failure modes of PINNs, and proposing regularisation methods based on the observations made. Namely, the optimisation difficulties that arise due to the PDE 'soft constraint', which requires regularisation to ease the complexity of the loss landscape. Wang et al. [2021] identifies a different mode of failure in PINNs related to back-propagation in loss functions with multiple terms, using learning rate annealing to address said failure.

Wang et al. [2022] presents another way of observing the failure modes of PINNs, by using a Neural Tangent Kernel that allows them to quantify the convergence rate of different loss components, which are found to vary from the total error convergence. Addressing the discrepancy by calibrating those convergence rates alleviates the issue, and they manage to provide not only an empirical but a theoretical justification for this loss weighting.

Anagnostopoulos et al. [2023] investigates how the weights within the model evolve during training, labelling two main stages that it correlates to information bottleneck theory. This opens up the possibility of exploring more on the nature of the stability of PINNs and neural operators.

2.2.4 Training developments

While the failure modes and underlying causes may differ between applications, they usually manifest during the training stage, where the loss function may become stuck in local minima (be this due to network capacity, complex loss landscape or inadequate training methods) leading to lack of convergence or inaccurate solutions

Krishnapriyan et al. [2021], as opposed to a lack of expressivity arising from the chosen network architecture. Training is thus one of the major challenges of PINNs, as well as machine learning more broadly. So far, we've examined how neural network design can improve accuracy and data efficiency by incorporating physics or through other built-in methods. However, these strategies do not directly address the optimisation algorithms or the training process itself. In this subsection, we cover research examples that show how the method in which the network is trained can also help to increase network performance.

Domain decomposition, which is also employed in other ML methods for solving PDEs [Li et al., 2020], has also been adapted for PINNs in Jagtap and Karniadakis [2020], and for a PINN/Deep Galerkin Method hybrid in Kharazmi et al. [2021]. Instead of optimising a network based on the full dataset, domain decomposition works on the principle of restricting the training data into subsets in space or time, and training sequentially over these subsets. This is often found to help simplify the optimisation landscape and improves convergence. Algorithms like [Jagtap et al., 2020b] take the concept further and also partition the neural network into subnetworks, creating potential for parallelisation in the future to reduce training costs.

Krishnapriyan et al. [2021] describes training issues stemming from complex loss landscapes introduced by the addition of soft PDE constraints, and proposes two solutions that address these failure modes. One of these is posing the problem as a sequence-to-sequence learning task, similar to the domain decomposition tasks. It also proposes curriculum regularisation, where training is first carried out for an easier problem, in order to find a good initialisation for the weights of the more complex problem. Both of these help achieve better errors, demonstrating the merits of the approaches. This shows that weight initialisation, beyond simply preventing issues with vanishing or exploding gradients, can also be used to improve the outcome of training. An example approach by Tarbiyati and Nemati Saray [2024] uses a coarse grid dataset to learn initial weights that can reduce the error of the PINN implementation, similar to the more recent FEM enabled pre-training for PINNs suggested by Wang et al. [2025b]. A last example by Khadijeh et al. [2025] presents a multistage PINN for multi-physics problems, where the training method consists of progressively increasing the complexity of the PDE that the network is trained on, contrasting with methods that attempt to minimise the entire problem from the start by including all PDEs in the loss function.

These approaches all help PINNs achieve enhanced accuracy with better computational efficiency, and serve to show that there are many areas where PINNs may

be optimised.

2.2.5 Adaptivity in PINNs

Given the wide range of possible implementations for PINNs and machine learning models in general, tuning the numerous hyperparameters for a specific application can be both challenging and time-consuming [Wang et al., 2023a]. Frequently, this selection is dependent on either prior knowledge or trial and error. The goal of implementing adaptive approaches then is to help change impactful hyperparameters in response to their performance throughout the training, in this way finding optimal parameters and applying them within a single process. Furthermore, adaptive strategies should also help with generalisability, as models can adapt when utilised for different problems.

Over time, a wide range of adaptive strategies have been proposed to improve the training of PINNs, adjusting everything from method-specific parameters to more universal hyperparameters like adaptive learning rates, which are built-in to optimisation algorithms. Building on the domain decomposition methods mentioned in the previous subsection, Hu et al. [2023]'s work extends the decomposition methods with an adaptive strategy that ideally arrives at more optimal sub-domain partitions to maximise efficiency. Activation functions can also benefit from adaptive selection as shown in Jagtap et al. [2020a], a concept similar to the subsequent Kolmogorov-Arnold Networks' learnable activation functions [Liu et al., 2025]. These networks, combined with PINNs in work by Rigas et al. [2024], combine the benefits of PINNs with learnable activation functions and an adaptive training scheme that enhances accuracy and generalisability.

Other notable contributions include self-adaptive loss weighting [Xiang et al., 2022], and adaptive selection of the loss function [Zeng et al., 2022]. The latter work also features adaptive sampling of collocation points (discussed in detail in the next section), which together with the above, allow the DNN to solve high-dimensional PDEs with improved accuracy and convergence speed. Finally, Gu et al. [2021] proposes a method for how training samples can be weighted to accelerate convergence speed and robustness of PINNs.

The papers discussed so far in Section 2.2 have addressed a wide range of developments that have broadly improved PINNs; however, none have focused on the relationship between the placement of collocation points and model accuracy. Among the various adaptive strategies discussed, the distribution of collocation points represents a particularly critical aspect, as they are a key defining feature of PINNs.

Given that, for a given problem the placement of collocation points can significantly influence training performance and solution accuracy, identifying more effective point distributions is a natural target for an adaptive strategy. This forms the central focus of this thesis, and in Section 2.3 we examine this topic in detail, exploring how the distribution of the set of collocation points affects the performance of PINNs, and how adaptive methods of sampling have been used to improve the overall capabilities of PINNs.

2.3 Adaptivity and Collocation Point Sampling

In PINNs, collocation points are the locations in the domain where the governing physics are enforced. This is achieved by evaluating the PDE residuals at these points and incorporating them into the loss function, as described in more detail in 1.4. Since the learning of the physical laws depends on these points, a poor selection of collocation points can result in regions of high error or inefficient training. This is especially true in unsupervised problems, where the loss function is exclusively defined through collocation points in the absence of data points.

Practically, even in low-dimensional spaces, we observe that for problems with important features located in small regions of interest require a higher density of collocation points for PINNs to train, for example, the inverse problem in Wang et al. [2023b]. For a given problem, we can thus pose that there exist some sets of collocation points that is better than others, or at least that some collocation points are more valuable during training. However, for an unknown solution, knowing this sampling *a priori* is not always possible. In addition, manual distribution of individual points can significantly increase costs, especially as problem complexity or dimensionality causes the required number of collocation points to increase. Adaptively selecting points can address these issues, and hopefully, the smart selection of a set of collocation points can also reduce the number of points needed to adequately cover a problem domain, reducing the computational cost of evaluating the loss function and its derivatives.

This problem of collocation point sampling is at the heart of the work carried out in this thesis, and so in this section, we comprehensively explore the different sampling strategies that have been investigated to maximise the efficiency of collocation points. Section 2.3.1 looks at indirect approaches that don't actually re-sample the chosen points, instead adjusting loss functions or creating weight functions to weigh differently the initially sampled points. The initial sampling of these points

is also important, and different initial sampling mechanisms are discussed in 2.3.2. The focus of the final section, 2.3.3, is the direct adjustment of the position of collocation points in the domain. This section considers the initial sampling, methods for selecting new points, and broader resampling strategies.

2.3.1 Point weighing

In this subsection, we discuss adaptive strategies that affect how a model trains from collocation points without directly changing the actual position of collocation points. This is primarily done via point weighing, which starts by quantifying which are 'good' points and biases the model to prioritise these. However, it's worth noting that most of these approaches don't use a uniformly random initial grid, but quasi-random grids that improve the initial spacing and avoid redundant points, such as Latin hypercube sampling (see Section 2.3.2).

In the Augmented-Lagrangian PINN [Son et al., 2023], the loss is discretised across the set of collocation points as usual, along with a Lagrange multiplier that itself is dynamically adjusted during training. Whilst the collocation points themselves aren't adjusted, these multipliers attached to them add a penalty to the output from each point with the aim of preventing instabilities due to the boundary conditions, which they prove to aid convergence.

Other methods employ adaptive weighting strategies to directly alter the contribution of particular points by assigning multipliers locally. The Self-Adaptive (SA) PINN introduced by McClenny and Braga-Neto [2023] follows this approach, formulating the loss minimisation as a min-max optimisation problem: maximising the weights and simultaneously minimising the overall loss. This work is followed up by DASA-PINNs [Zhang et al., 2023] where a Differential Adversarial (DA) sub-network is trained to predict the optimal weights. A different method not reliant on a separate sub-network is proposed by Anagnostopoulos et al. [2023], which uses a residual-based attention (RBA) scheme to scale the overall loss function that defines the contribution of different collocation points. The general method achieves relative error of order 10^{-5} on general benchmarks, but furthermore is also validated on some practical application benchmarks in Ramirez et al. [2025], performing better than SA-PINN and vanilla PINN.

Chen et al. [2025] presents a more hybrid approach for solving coupled Allen Cahn and Cahn-Hilliard equations. In this approach, a similar loss weighting framework is employed, but based on computation of a Neural Tangent Kernel (NTK) matrix. This is supplemented with a random-batch method that randomly selects

representative points rather than calculating weights for the entire set, which eases the burden of computation. Similar flexible point weighting methods are used in a generative adversarial / PINN approach in Li et al. [2022], where the weights are updated at every training step. A more sophisticated approach for weight computation is developed in Gu et al. [2021], where the self-paced learning framework consists of a secondary network trained to select more challenging points later in the training of the solver model. Despite the additional cost, this approach again helps outperform basic models on convergence speed and robustness.

Most of the above models utilise the residual errors directly or indirectly as the measure of what collocation points or data samples are the most challenging, and thus should be weighted more heavily, with the main differences between models being how this information is used to then weight the collocation points. However, an important question is how those points are distributed to begin with, which is the topic of the following section.

2.3.2 Initial Point Sampling

Before moving to the direct movement of collocation points, which we will term 'resampling', we will discuss the initial sampling of points before training begins. As we briefly mentioned in the previous subsection, quasi-random sampling methods such as Latin Hypercube Sampling (LHS) [McKay et al., 1979] are often used instead of uniform grids and random distributions, as quasi-random methods add stochasticity whilst ensuring good spacing of points. This is the main advantage over uniform random sampling; that every area of a domain can be covered without any redundant points being placed too close to one another. As for the sampling methods themselves, there are various: Hammersley sequencing [Hammersley and Handscomb, 1964], LHS [McKay et al., 1979, Stein, 1987], Halton Sequence [Halton, 1960] and Sobol Sequence [Sobol, 1967].

A thorough comparison between all of the above non-adaptive sampling methods and uniform resampling in PINNs is done by Wu et al. [2023], which also considers using various random resamples generated every certain number of training iterations. In the comparison we see that across the original benchmarks, all of these initial sampling improvements provide noticeable benefits to accuracy over the original PINNs. These observations suggest that, for a given problem, certain collocation point distributions are inherently more effective than others. By itself, however, a good initial sampling strategy lacks the ability to identify, or still less adapt, to regions of physical interest as training develops. If these uninformed, fixed distribu-

tions can provide benefits, then methods that could leverage information gathered during training and adjust collocation points accordingly could be even more effective, motivating research into adaptive resampling techniques.

2.3.3 Point Resampling

In this subsection, we cover the direct placement and movement of collocation points throughout training in response to the current model's performance. These approaches begin from initially random or quasi-random sets of points. We first distinguish methods that work through 'refinement', where these original sets of points are retained, and new points are added in 'difficult' areas of the domain to help with training. This kind of adaptive refinement is proposed in Lu et al. [2021b], where the points are added in areas where the residual is highest (in a greedy algorithm), in a method they thus name Residual-based Adaptive (greedy) Refinement, RAR. An improvement to the greedy strategy, proposed by Nabian et al. [2021], uses a probability distribution proportional to the loss, which they refer to as the importance sampling method.

From this point, however, many approaches were developed in parallel to propose different formulations for a Probability Distribution Function (PDF). In addition to the refinement strategy, we also observe replacement to become more commonplace, where points previously trained on are not kept and instead the entire set of collocation points is replaced to adapt to the current model's strong and weak spots. In this section we cover a number of relevant resampling approaches, which share many similarities. Out of these, the PDF proposed in Wu et al. [2023]'s RAD (Residual-based Adaptive Distribution) is one of the most general. This is due to the implementation of two hyperparameters that affect the probability $P(\mathcal{X})$ of a given point being chosen (see equation 2.1): c, adjusting the base probability; and k, adjusting the scaling of the function Y. In RAD, $Y(\mathcal{X})$ is taken to be the point-wise PDE residual, but can be any point-wise metric. The term $\overline{Y(\mathcal{X})^k}$ denotes the mean value of $Y(\mathcal{X})^k$ over the candidate sampling set, providing a normalization factor. These hyperparameters are manually selected, and certain combinations recover the probability distribution functions proposed in other approaches.

$$P(\mathcal{X}) = \frac{Y(\mathcal{X})^k}{\overline{Y(\mathcal{X})^k}} + c \tag{2.1}$$

For example, the PDF in Tang et al. [2023] is a case where the probability scales with the squared loss, equivalent to a hyperparameter of $k = 2$ in RAD. However,

the broader implementation is unique in that the final PDF is predicted through the use of a generative model, KRnet Tang et al. [2020]. Other approaches like Gao et al. [2023]'s 'failure-informed' resampling again use a similar PDF framework and function to select new points. In this specific case the difference is the lack of a term to add base probability (equivalent $c = 0$), which means points are only added to areas of significant loss.

The approach itself can vary in many ways beyond how the PDF is defined. In Zapf et al. [2022], the PDF employs a simple greedy refinement strategy, but the addition of new points is combined with removal of points with lowest residual errors, keeping the total number of collocation points, and thus the computational costs, fixed. The approach by Daw et al. [2023] proposes a similar approach, replacing less useful points with random points, in a hybrid refinement-replacement approach that attempts to mitigate stalling due to poor sampling of points. A third example of these changes is found in Subramanian et al. [2023], where their 'adaptive self-supervision' algorithm combines a progressive annealing refinement scheme with automatic restarting of training when a plateau in optimisation loss is detected, where the set of points is reset to a uniform random grid. This progressive refinement slowly adds refined points to an initially uniform grid in order to adjust the balance between local adaptivity and global collocation point coverage as training progresses.

One downside of not assigning a base probability to all candidate points in the PDF is that an adaptive strategy might create clusters of overlapping points that overly bias the model. However, design of the PDF isn't the only way this can be addressed. In Subramanian et al. [2023], we've noted how the initial grid is uniform, and the points reset to that uniform grid periodically, which ensures that general information of the entire domain isn't lost. Hou et al. [2023] deals with the same challenge by separating the pool of collocation points into a fixed set and a movable set, with the latter points changing based on an error estimator. Additionally, they also implement loss weighting based on other factors to bias the importance of the points across both sets. Chen et al. [2025], mentioned in the previous sub-section for its adaptive weighting method, also contains advanced resampling methods. Like Hou et al. [2023], they separate collocation points into an adaptive set that moves based on residual errors, and a fixed grid. They however maximise the efficiency of the fixed grid points by implementing random translations that allow the network to gain new information while ensuring broad coverage of the domain is kept by retaining a uniform grid.

A similar idea of distinguishing different types of collocation points for resam-

pling is proposed in Hanna et al. [2022], where different groups of points are defined targeting different equations in systems of many PDEs. For this approach, the PDF uses different loss terms to guide the resampling of the different groups of points, ensure points are distributed to target the learning of all relevant governing equations. A second novelty is that the network itself is also divided into sub-networks that are trained separately to predict the different physical outputs (e.g. velocity, pressure, etc...), which is seen to help reduce the generalisation error in 1D and 2D two-phase flow examples.

An important note is that the adaptive resampling work described so far in this subsection has focused exclusively on the sampling of collocation points, and thus on unsupervised problems (taking into consideration only physics data and not the availability of solution data). As the purview of PINNs also includes problems where solution data points are also available, a remaining factor to consider is how these adaptive strategies can adapt to the influence of data points. This is addressed by Lau et al. [2024], which proposes a methodology to jointly consider the selection of experimental and physics (collocation) points, which allows us to expand the benefits of adaptive resampling to inverse and transfer learning problems.

Except for Nabian et al. [2021], the methods we have discussed so far share one broad approach requiring the introduction of a dense set of candidate points. This dense set acts as a pool of potential collocation points that covers the entire domain. Although only the selected subset of collocation points will be used during training, each time resampling occurs the candidate set is regenerated and the local PDE residuals (or other error indicators) are evaluated at every candidate point in order to identify the new set of collocation points. As this pool of candidate points must adequately cover the domain, it contains a significantly greater amount of points than will be chosen, and so processing the entire set can be computationally expensive in terms of both memory and runtime. However, this is not the only way adaptive resampling has been tested. A few other papers attempt using relatively smaller number of point 'seeds' where the calculated loss is taken to represent the neighbouring region as drawn by Voronoi tessellation. Again using a PDF, areas with higher losses are assigned more points, except rather than using the exact seed locations collocation points are randomly added to sub-domains neighbouring those seeds. In Nabian et al. [2021], this is done for computational efficiency benefits, although it clearly carries some discretisation error depending on the number of seeds chosen, and a non-optimal placement within those subdomain regions. That being noted, similar sub-domain adaptive sampling implementations of that approach

in Zeng et al. [2022] achieves relative errors on the order of $O(10^{-4})$ in high dimensional problem solving, comparable to other PINNs methods. More recently, a hybrid sampling approach proposed by Jin et al. [2025] merges a uniform sampling approach with this type of area-refinement adaptive sampling (similar to Hou et al. [2023] and Chen et al. [2025]) to moderate success in forward solving of stationary Fokker–Planck–Kolmogorov equation. Most recent updates to this methodology can be seen in Yang et al. [2023] and Feng et al. [2025], demonstrating the effectiveness through more benchmarks.

We can also consider the perspective of resampling as a problem that requires finding an ideal set of collocation points that maximises accuracy, with a secondary objective of minimising the number of collocation points required. Posing the resampling problem as a minmax approach, we can employ a neural network to solve the task of finding this optimal distribution of points. A few papers have been published considering very different neural networks approaches to guide adaptive resampling. Tang et al. [2023] and Wan et al. [2025] utilise a generative model to approximate a PDF; Anticev et al. [2024] uses graph models to cluster points and identify areas based on errors and robustness; Wang et al. [2025a] uses a network to model the residual indicators continuously, precisely identifying the areas of importance for adaptive resampling; and the last example of Gao et al. [2025] integrates a ResNet to move the points dynamically to great benefit relative to other PINN approaches in complex nonlinear and high-dimensional finance applications.

Across all the methods we have described that seek to move (or add) collocation points, a commonality is the almost exclusive use of the residual as the input information, regardless of whether this information is simply used in a static PDF or fed to a trained network. We can thus highlight a gap in the input to the resampling process. Considering traditional CFD, adaptive meshing targets sharp gradients or high curvatures in solution variables to indicate regions for potential refinement (velocity or pressure gradients can, for example, indicate the position of shock fronts, separation, and other important physical flows).

Whilst a lot of the initial research on adaptive resampling uses the paradigm of evaluating a PDF for a large number of candidate points, this itself is not very scalable to more complex / higher dimensionality problems. As this could undermine the potential benefits to scaling of using adaptive resampling (which can reduce the number of collocation points needed to cover a domain), there is also a need for research into alternative methods of guiding adaptive resampling. This leaves a number of questions to answer, from quantifying the practical usefulness and

scalability of PINNs and adaptive resampling to developing and testing new methods for adaptive resampling.

Additionally, it is worth noting that adaptive sampling exists not only in PINNs but also in some of the other ML approaches like the Deep Galerkin method Aristotelous et al. [2023] and Deep Ritz Wan et al. [2025], and many of the methods and techniques we have explored in this review, and will introduce in this thesis, have the advantage that they may be useful generally across any ML methods relying on point-based training.

Existing adaptive resampling methods therefore rely on a narrow information source. The present thesis addresses this by investigating whether alternative indicators inspired by traditional mesh refinement criteria, or derived from higher derivatives of PINN residuals, can be used to guide resampling more effectively. The scalability of these methods is also examined, together with refinements to the adaptive resampling methodology itself.

Chapter 3

Sampling the Landscape

In this chapter, initial background work carried out before the introduction of our own contributions in Chapters 4-6 is presented. Ground truth solutions are generated via finite difference methods and later used to assess accuracy. A simple test of the effect of collocation point distribution is then set up, in which points are manually biased and the resulting change in model accuracy is observed. This short test shows that positive changes to model error can be obtained by adjusting the placement of collocation points. Finally, two methods of residual-based adaptive sampling are validated and compared to the use of a fixed quasi-random distribution for solving a PDE.

The chapter begins with a brief introduction explaining the role of this work in the context of the thesis. The methods section then details the problem considered, the generation and validation of a sample solution, the vanilla PINN model used and key parameters selected, and the adaptive resampling/refinement methods from literature we reproduce. The results section describes the experimental strategy and findings, followed by a discussion section that provides an analysis of the outcomes.

3.1 Introduction

As discussed in the literature review, Physics-Informed Neural Networks (PINNs) and machine learning approaches have become increasingly compelling alternative method for solving complex physical problems. A significant amount of research has focused on improving the training process of PINNs, with adaptive sampling of collocation points emerging as a particularly promising method of improving PINN performance. In the literature review, a gap is identified in how adaptive sampling is performed and in the type of information used to guide it, which motivates the investigations presented in later chapters.

However, numerous benchmarks are available to consider, along with many different possible approaches to the sampling problem. In this chapter the chosen benchmark, Burgers' Equation, is presented, and a ground truth solution is generated and validated to test the PINN model on.

The importance of collocation point selection is then demonstrated by manually biasing the point distribution toward the region of physical significance within the benchmark problem. This experiment shows that the positioning of collocation points impacts performance, which motivates the development of resampling techniques in subsequent chapters. Finally, we attempt to recreate the results of utilising adaptive resampling techniques from the literature to bias points in the same problem. These results also serve as baseline to compare against the resampling techniques we develop in subsequent chapters.

3.2 Methods

3.2.1 Benchmark Problem: Burgers' Equation

The benchmark we will use in this chapter and Chapter 4 is the viscous 1D Burgers' Equation, with one spatial dimension and a temporal dimension, given by:

$$uu_x + u_t = \nu\, u_{xx}, \quad x \in [-1, 1], \quad t \in [0, 1], \tag{3.1}$$

where the magnitude of ν determines the relative effect of diffusion. Dirichlet boundary conditions are assigned:

$$u(-1, t) = u(1, t) = 0,$$

as is the following initial condition:

$$u(x, 0) = -\sin(\pi x).$$

Burgers' Equation is a common example in the literature for PINNs but also a classic fluid dynamics problem for which ground truth can be obtained easily with finite differences. More importantly, at low values of ν and with appropriate initial conditions, a sharp jump in velocity occurs (which we refer to as 'shock' as the viscous term prevents an actual discontinuity from forming). The challenge of accurately computing this phenomenon makes it a good test for numerical models of any kind, especially as ν is made smaller and the 'shock' is more pronounced, requiring

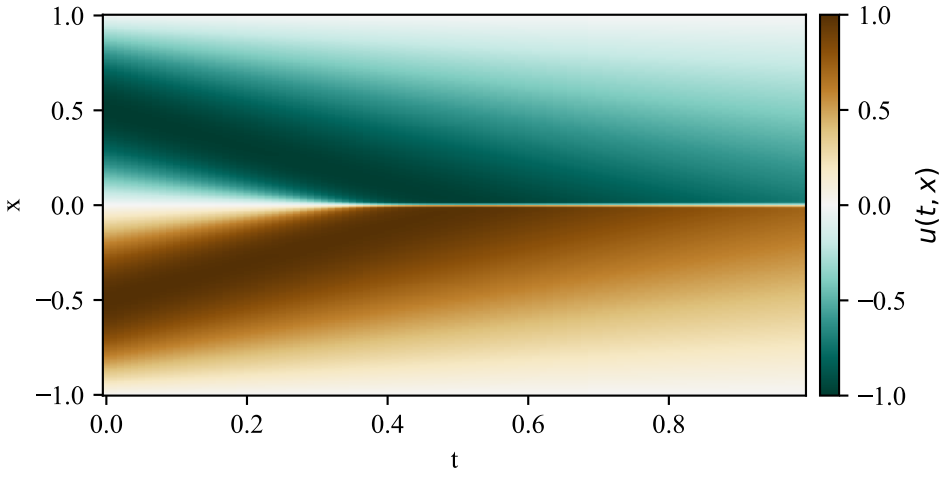

Figure 3.1: Reference solution to 1D Burgers' Equation, with $\nu = \frac{0.01}{\pi}$

computationally expensive fine meshes from traditional methods. Additionally, the 'shock' feature is an easily identifiable area of complexity and, with symmetrical initial conditions (like in the example considered above), it is perfectly horizontal. This makes the problem very suitable for our point-biasing experiment, as it is easy to manually place points around the shock location.

3.2.2 Generating Ground Truth Solutions via FDM

To obtain reference solutions, the governing equation is reformulated as an ordinary differential equation and discretized using a central difference scheme. The implementation is carried out in `SciPy`, iterating over the fixed computational domain $-1 < x < 1$ and $0 < t < 1$. The spatial and temporal discretisations are chosen such that $N_x = 2N_t$, where N_x and N_t denote the number of grid points in space and time, respectively.

A grid refinement study is performed by progressively doubling the resolution in both dimensions, starting from $N_x = 200$ up to $N_x = 6400$. For each refinement level, the solution obtained on the finer grid is compared to the previous coarser grid by restriction, and the resulting difference is used as an error measure. Figure 3.2 presents these results, where the vertical axis shows the computed error and the time taken in minutes, whilst the horizontal axis represents the number of spatial grid points N_x. Note that the temporal resolution is scaled proportionally, with $N_t = N_x/2$.

The time required to compute each grid grows nonlinearly with number of grid points. However, the discrepancy between the two finest grids tested is on the order of 10^{-8}, which is several orders of magnitude more accurate than what is typically expected of PINNs. The finest grid is therefore deemed sufficiently accurate for

the purposes of providing a ground truth solution against which to compare model performance on the 1D problem.

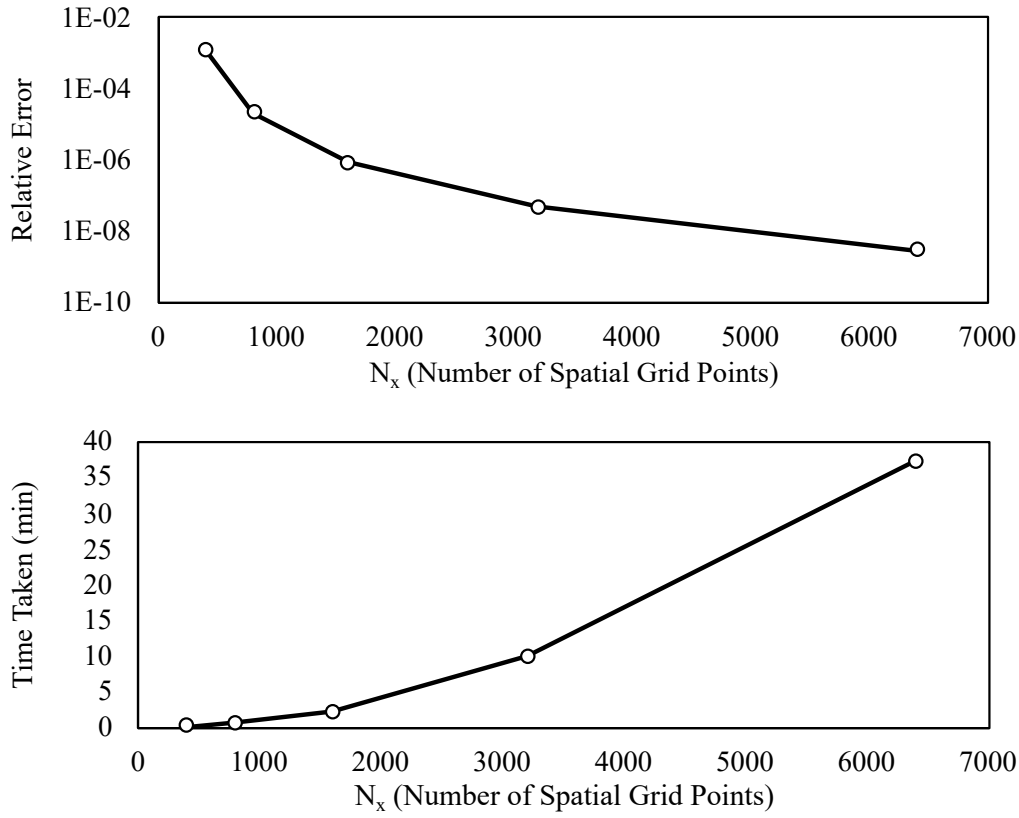

Figure 3.2: Grid independence study of FDM solution. Above, relative error (calculated as difference with previous grid) is plotted against number of points in the spatial grid. Below, time to compute solution is plotted against number of points in the spatial grid.

3.2.3 Vanilla PINNs model and manual point biasing

In this subsection, we will describe a baseline PINN model following the formulation described in Raissi et al. [2019]. This model is used to test whether manually adjusting the position of collocation points can help performance, and the method describing this point biasing is described.

The architecture consists of a fully connected neural networks with 3 intermediate layers, with nodes using the hyperbolic tangent (tanh) activation functions. Training is performed using the L-BFGS optimiser, and the loss function is defined as the sum of three terms corresponding to the PDE residual, initial condition and boundary condition. Due to the stochastic nature of ML training due to variance in both the network initialisation and the optimisation process, multiple repeats with different seeds are carried out for each case investigated in this section as well as in subsequent chapters of this thesis. Reported errors therefore represent the mean across runs,

with variance and notable outliers discussed explicitly discussed where relevant.

In order to test the placement of points, and knowing *a priori* the discontinuity in our 1D Burgers' Equation problem lies along the $x = 0$ axis, the decision is made to increase the number of points in a narrow region around $x = 0$. The domain is divided into 3 horizontal subsection within which points are distributed randomly. To test the bias, the total number of collocation points in the domain N is kept constant, with the numbers within each region changing according to a manually selected bias ratio.

The bias ratio is quantified as the fraction of total points in the middle region normalised by the area of the middle region as a fraction of the whole domain, as shown by 3.2. This definition allows easier interpretability of the bias, where 1 corresponds to an unbiased allocation of points, while a value of 2 corresponds to twice the expected density. An example of this simple bias is shown in Figure 3.3.

$$\text{Bias ratio} = \frac{N_{Bias}/N_{Total}}{A_{Bias}/A_{Total}}, \tag{3.2}$$

Where:

$$\begin{aligned} N_{Bias} &= \text{Number of points in biased section,} \\ N_{Total} &= \text{Total number of points,} \\ A_{bias} &= \text{Area of biased section,} \\ A_{Total} &= \text{Total Area} \end{aligned}$$

3.2.4 Validation of residual based adaptive sampling methods

The work of Wu et al. [2023] is used as the methodological baseline against which we will compare other adaptive resampling methods in future chapters. In this section, their proposed model, parameter choices and training procedure are described. An effort is made to recreate these methods and validate their performance, the results of which are presented later in subsection 3.3.2.

The model itself is a fully connected neural network with 3 intermediate layers, using the tanh activation function, consistent with the previously described vanilla PINN architecture. The main distinction lies in the approach to training strategy and optimiser selection. A hybrid optimisation strategy is used, where training is initialised with the Adam optimiser and then switched to L-BFGS after a previously selected number of iterations is reached. When the training set of collocation points

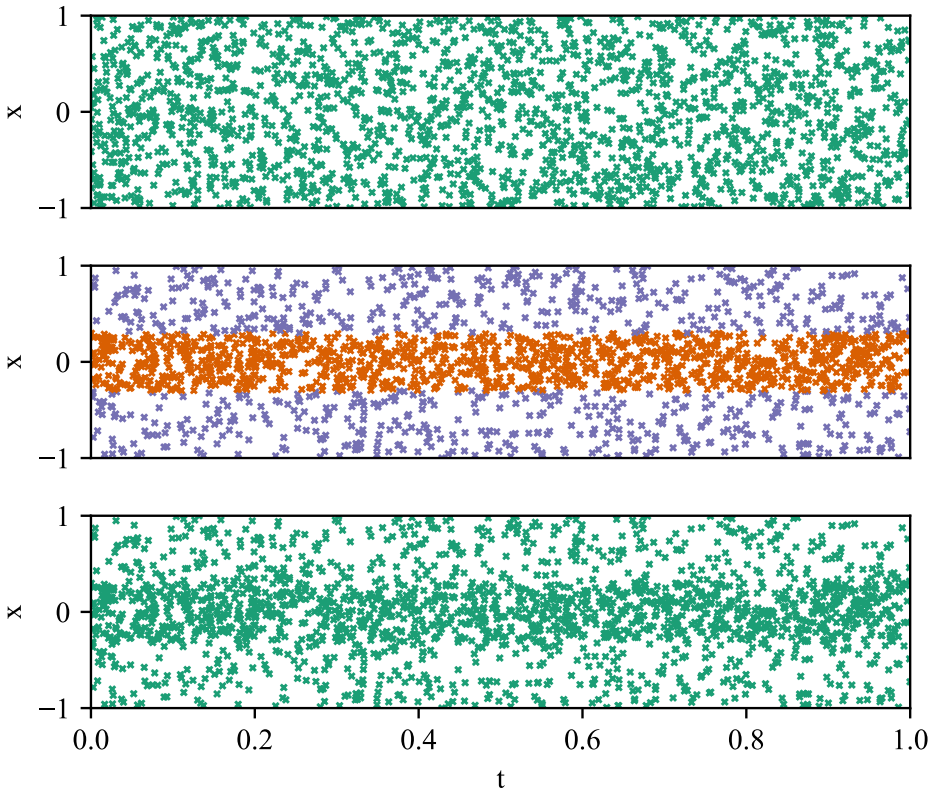

Figure 3.3: Illustration of a uniform random distribution with no bias (above) and the simple bias around the $x = 0$ axis (middle and bottom) using bias ratio 1.5. $N = 2{,}000$ for both cases.

is changed either by refinement (addition of points) or replacement, the optimiser is reset to Adam for 1,000 iterations before switching back to L-BFGS.

An output transformation is also applied to the output layer, in order to enforce initial and boundary conditions in a 'hard manner'. This transformation is of the form:

$$output = -\sin(\pi * x) + (1 - x^2)ut$$

Where u denotes the raw network output. Here, the function manipulates the raw output so that the final output automatically satisfies the conditions described in subsection 3.2.1.

The method for both resampling and refinement of collocation points chooses the next set of collocation points $\mathcal{T}$ from a dense set of random points $\mathcal{X}$ covering the domain. All points in $\mathcal{X}$ are assigned a probability ($P(\mathcal{X})$) according to 3.3, and the prescribed number of collocation points are chosen according to the normalised probability distribution function $\hat{P}(\mathcal{X})$ given by 3.4:

$$P(\mathcal{X}) = \frac{Y(\mathcal{X})^k}{\overline{Y(\mathcal{X})^k}} + c \tag{3.3}$$

$$\hat{P}(\mathcal{X}) = \frac{P(\mathcal{X})}{\|P(\mathcal{X})\|_1} \tag{3.4}$$

Here $Y(\mathcal{X})$ is the vector containing the magnitude of the information source chosen (e.g. the current residual or its derivatives, or some derivatives of the current estimate of the solution), $\overline{Y(\mathcal{X})^k}$ is the mean value across all points in $\mathcal{X}$ and k, c are constant hyper-parameters that affect the resampling behaviour.

The first adaptive sampling method proposed by Wu et al. [2023] is Residual-based Adaptive Resampling (RAD). Here, the previously described architecture and training strategy is employed, and residual information is used in equation 3.3 to calculate the probability distribution function (PDF) at every point $\mathcal{X}$. In RAD, once this is done the previous set of collocation points is discarded and wholly replaced by N new collocation points according to the probabilities of PDF. The entire process is described in algorithm 1:

Algorithm 1 RAD

1: Sample N initial residual points using a uniform random sampling method;
2: Begin training for a fixed number of steps using Adam optimiser;
3: Switch to L-BFGS for 1,000 optimisation steps;
4: **repeat**
5: Replace all collocation points with N new points selected from candidate set using Eqs. (3.3) and (3.4);
6: restart training with Adam, then switch to L-BFGS;
7: **until** Predetermined number of resampling loops is reached

The second proposed algorithm, Residual-Based Adaptive Distribution with Distribution (RAR-D), described in algorithm 2, aims to continuously add new collocation points instead of discarding old collocation points. Consequently, the training strategy involves starting with a lower number of collocation points, and at every sampling stage adding a set number of collocation points according to the probabilities calculated using equation 3.3 (again according to the local residual information at every point in $\mathcal{X}$). Training then continues until the number of collocation points reaches the predetermined N.

Algorithm 2 RAR-D

1: Sample N_0 initial residual points using a uniform random sampling method;
2: Begin training for a fixed number of steps using Adam optimiser;
3: Switch to L-BFGS for 1,000 optimisation steps;
4: **repeat**
5: Select new points from candidate set using Eqs. (3.3) and (3.4);
6: Add N_{new} new points, increasing its size from N_i to $N_{i+1} = N_i + N_{new}$;
7: Restart training with Adam, then switch to L-BFGS;
8: **until** Number of collocation points reaches specified amount, $N_i = N$

Additionally, the two adaptive resampling methods themselves are compared to

a fixed set of quasi-randomly distributed points. The purpose of this comparison is that quasi-random algorithms avoid redundant clustering of points, and eliminate the possibility of large regions of the domain receiving receiving no coverage. In the tests presented below, we specifically use the Hammersley sequence Hammersley and Handscomb [1964], identified as one of the best performing quasirandom sequences in Wu et al. [2023].

3.3 Results

In this section, we describe the results of the experiments we've described in the previous section. First, for a vanilla PINN model we present the error as manual bias is introduced into the collocation point set random distribution. Next, for the validation of the adaptive sampling method of placing collocation points, we present a table comparing the error obtained in this thesis against the errors quoted in the literature using the same architecture and parameters.

To measure and compare the accuracy of the different methods considered an L_2 relative error metric is used at the end of training. This compares the prediction u to ground truth u_{gt} as follows:

$$L_2 = \frac{\sqrt{\sum (u(i) - u_{gt}(i))^2}}{\sqrt{\sum u_{gt}(i)^2}} \, .$$

Note that the sums in this last expression are over every point i in a very fine background grid which contains many more points than the number of collocation points used to evaluate u. Furthermore, a minimum of 20 repeats with different seeds to account for the stochasticity in machine learning optimisation, with the quoted L_2 errors always representing the average error across these repeats.

3.3.1 Manual point biasing

Figure 3.4 shows the results of the simple point biasing described in section 3.2.3, plotting the solution error obtained by the model against different levels of bias ratio (see Eq. 3.2), where a bias of 1 represents a uniform random distribution of points. For ease of visualisation, this is plotted as a horizontal dashed line to better allow for comparison. These results are plotted for models using a high number of collocation points $N = 8{,}000$ and low number of collocation points $N = 2{,}000$. For both low and high N, error can be observed to decreased as bias increases above 1.5, with the effect being much more significant for the $N = 2{,}000$ (decreasing from

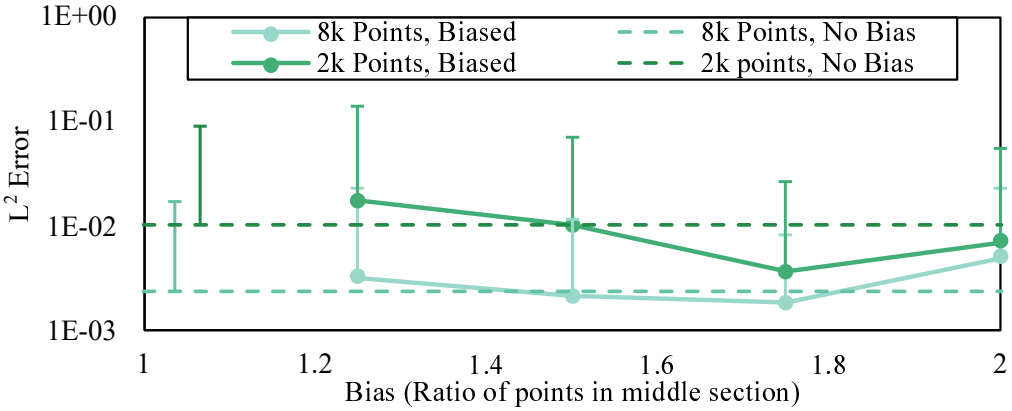

Figure 3.4: L_2 Error against bias, defined as ratio of points assigned to the section containing the Burgers' Equation discontinuity, for two different N. Error bars represent ± 1 standard deviation (SD).

1.03×10^{-2} to 3.63×10^{-3} with a bias of 1.75). In the higher N case, with enough points to appropriately cover the domain, only light improvements are observed, and too high a bias actually increases the resulting error. The standard deviation is greater than the mean due to a number of simulations across all cases failing to converge. These results both show that the distribution of collocation points, even whilst still being sampled randomly within each subsection, directly affects the performance of PINNs, although the crude approach is highly inconsistent as seen by the high variance across all cases.

3.3.2 Validation

Whilst Wu et al. [2023] tests their adaptive sampling methods across a range of hyperparameters, we compare against the main results they present. These use 2,000 collocation points and optimised hyperparameters: $k = 1$, $c = 1$ for RAD, and $k = 2$, $c = 0$, adding 10 points at a time, for RAR-D. As well as comparing several quasirandom algorithms of fixed points, their study also considers a uniform random resampling strategy. This strategy is not considered in this thesis, as its performance is consistently inferior to RAD in their benchmarks and it is not as representative of the standard fixed-grid baseline used in the literature. As a result, we reproduce and compare the fixed quasirandom Hammersley case, the RAD case and the RAR-D case.

The results of this comparison are shown in Table 3.1. Using $N = 2{,}000$ and approximately 50 repeats, we reproduce similar errors for the fixed quasirandom grid case, and record a higher error around the same order of magnitude using the RAD algorithm. In contrast, the model does not converge to a solution when using the RAR-D method. Despite the potential benefits to cost of beginning training

with fewer points, the implementation of refinement introduces additional parameter choices (refinement amount, maximum number, etc) without a clear benefit to error across other benchmarks. As a result, it is not pursued further in this thesis over complete collocation point set replacement. RAD is significantly more accurate than using random uniform sampling at low N, confirming the advantage of adaptive sampling when the number of collocation points is limited.

	Quasirandom	RAD	RAR-D
[Wu et al., 2023]	3.02E-2 ± 2.98E-2	2.00E-4 ± 0.00E-2	3.00E-4 ± 0.01E-2
Validation	3.12E-2 ± 1.93E-2	7.81E-4 ± 6.88E-4	2.02E-1 ± 1.01E-1

Table 3.1: Table showing the L_2 error of different sampling methods cited in Wu et al. [2023] and reproduced in this thesis, using $N = 2{,}000$ collocation points. Values are reported as mean error ± one standard deviation.

3.4 Discussion

The aim of this chapter is to set up the method of PINNs and adaptive sampling that the contributions of this thesis will be based on, and to show the effect on model performance of collocation point placement. The first benchmark problem to be examined, 1D Burgers' Equation, is described, along with the generation of the ground truth solution that is used. A simple test is carried out to show the impact that strategic positioning of collocation points can have on model accuracy, by biasing the number of collocation points assigned to a horizontal section around the discontinuity in the 1D Burgers' Equation solution. More sophisticated methods from the literature for positioning these collocation points are then examined: a quasirandom algorithm that improves on the uniform random distribution in vanilla PINNs without any further changes to the training strategy; and adaptive sampling, which utilises a probability distribution function and information obtained throughout training to select new collocation points. The quasirandom fixed set approach and the RAD approaches from Wu et al. [2023] are reproduced, showing a distinct advantage to using adaptive sampling methods with a fixed number of collocation points.

The results of manual bias with $N = 2{,}000$ show relatively high errors (compared to adaptive sampling) of around 10^{-2}, improving to up to 3.6×10^{-3} by biasing points towards the section containing the discontinuity. However, this error is higher than the error obtained by increasing the number of collocation points to $N = 8{,}000$. For this N, the biasing is seen to also affect error, albeit less significantly. Whilst the network is not expected to be capable of reaching low error, as the approach is still

limited to uniform random distribution and no optimisation of model parameters is made, it still serves as a good example for showing the effect of collocation point distribution throughout the domain. It is observed not only that adding bias can decrease the error in scenarios with limited N, but also the opposite: how increasing the bias by too much can also have a negative effect, as in the case of $N = 8{,}000$ and a bias of 2.

We are not able to validate the refinement technique RAR-D, as the model is unable to converge on a solution when using this approach. Due to this, and the method's inferior performance relative to RAD across other benchmarks as reported in Wu et al. [2023], further research into this particular method of adaptive resampling is not continued in this thesis. However, the adaptive RAD and the fixed quasirandom strategy are both replicated. Errors of the same order of magnitude as reported by Wu et al. [2023] are obtained. Adaptive resampling is found to be 2 orders of magnitude more accurate than using the non-adaptive quasirandom set of points. Whilst these two results are limited to only the $N = 2{,}000$ case, these comparisons are extended further in the following chapters, where the quasirandom case is the baseline against which adaptive resampling methods are compared for a range of N and for different variations of the benchmark. In this section, however, the short comparison serves as suitable validation of these methods' feasibility.

Although the adaptive sampling and manual biasing approaches use different PINN formulations (Sections 3.2.3 and 3.2.4), including differences in boundary-condition enforcement, we can observe that manually biasing points through *a priori* knowledge towards the discontinuity does not reduce errors to the same extent as adaptive sampling. The latter can be more precise in the placement of points, as well as adapt throughout training in response to how the areas of high residual loss change due to optimisation of the network. In combination these findings can be seen to fulfil the aim of this chapter, namely to demonstrate the importance of collocation point location and examine an adaptive point sampling implementation that can select those locations appropriately. The next chapter examines a novel improvement to the adaptive sampling approach, focusing on the use of guiding information other than residual, inspired by mesh refinement techniques in other classical methods such as CFD.

Chapter 4

Guiding Information in 1D

This chapter describes the selection and implementation of novel information sources, which we term guiding indicators, into the adaptive resampling method for PINNs, and assesses their effectiveness.

First the role of our work in the broader literature is introduced, and the aims thus set to address the identified gap. The specific methods employed to carry out the work are then described. Lastly, the experimental strategy we designed is presented, and the results obtained are shown and then discussed.

4.1 Introduction

A key component of the PINNs methodology is the collocation points that provide the physics information to the network. These are universal regardless of whether the architecture is solving a forward or inverse problem, in both a supervised or unsupervised manner. In the previous chapter, results are shown for an unsupervised forward problem, where it is demonstrated that the selection of these points has an important effect on the training of a PINN. This agrees with contemporary results published by other researchers [Wu et al., 2023, Subramanian et al., 2023] - however, the process of adaptively selecting collocation points was limited to utilising residual information in their work.

Outside of machine learning, mesh-based numerical methods also benefit from targeted refinement in regions where important physical features are expected. In fluid dynamics applications, *a priori* knowledge of the flow can be used to bias the mesh toward boundary layers and zones where turbulence or shocks are expected, ensuring enough resolution is available to accurately capture these phenomena. There are also adaptive mesh-refinement methods that have been developed for, and applied to, these traditional CFD applications [ANSYS Inc., 2024]. In these

approaches, a low accuracy solution is initially obtained using a uniform coarse mesh for the discretisation, in which features such as shocks may be detected by measuring the highest velocity gradient cells. The mesh can then be increased in size, by refining select cells and thus allowing for higher resolution and ultimately higher accuracy.

Inspired by these traditional approaches, an investigation was undertaken to examine how analogous principles could be applied in the context of Physics-Informed Neural Networks, a mesh-less point-based method. In particular, rather than relying solely on residual-based criteria to drive adaptive resampling, alternative sources of guiding information were explored. These included quantities derived from the model's predictions during training - such as velocity gradients - as well as higher-order information such as residual derivatives, made accessible through automatic differentiation. The motivation was to determine whether these different indicators could be used to identify regions of interest more effectively than the residual alone.

The aim of this chapter is thus to compare these different sources of guiding information. Specifically, whether the proposed alternative methods for adaptive resampling could reduce the number of collocation points needed to achieve a given accuracy, or alternatively, improve accuracy when using a similar computational budget. The approaches were first applied to the one-dimensional Burgers' equation, where the region of interest is well understood, before extending the investigation to other configurations and an additional PDE, in order to evaluate the generalisability of our observations.

4.2 Methods

4.2.1 Benchmark problems

The main benchmark considered is the time-dependent, 1D, viscous Burgers' Equation described in the previous chapter. However, as part of our aims we also extend the work beyond the initial problem. This is done first by investigating changing PDE parameters and initial conditions for the 1D Burgers' PDE, and then by examining a second PDE, the Allen-Cahn equation.

Changing the PDE parameters and initial conditions in the Burgers' Equation benchmark significantly changes the solution, as it affects the location, magnitude and direction of travel of the nonlinear 'shock' that we posit may be easier or harder to learn depending on the distribution of collocation points. The effect of these changes on the solution can be clearly observed in figures 4.1 and 4.3. By considering

different viscosity and initial conditions, we want to investigate both whether the adaptive resampling methods can still adequately bias points towards those areas of interest, and whether this adaptive placement helps the model train more efficiently than with a fixed grid of points.

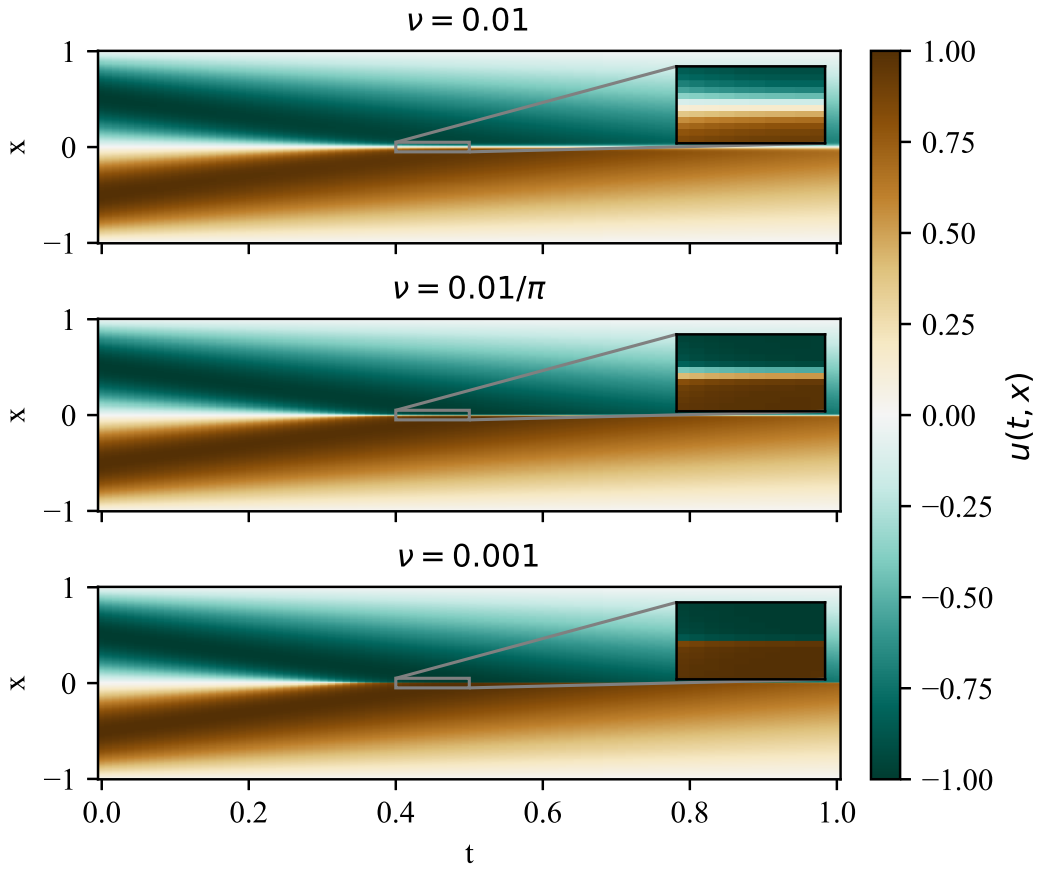

Figure 4.1: Reference solutions to Burgers' Equation for different ν, values exceeding display range are clipped to 1 and -1.

To still allow for the solution at the boundaries to be enforced in a hard manner, an output transformation was again used. As described previous in Section 3.2.3, this approach requires that these alternative initial conditions are chosen such that the transformation can be defined analytically to satisfy the constraints exactly. For this reason, Dirichlet boundary conditions are retained, whilst the new initial conditions are constructed using combinations of sine functions (Equation (4.1). The only requirement in the selection of these functions is that at $t = 0$, when the term enforcing the Dirichlet boundary conditions vanishes, each individual wave component satisfies $u(x = \pm 1, 0) = 0$. This ensures that the initial condition is smooth, analytically tractable, and flexible enough (by varying amplitudes and frequencies) to produce a range of solution behaviours. The general form of such an initial condition

is given by Equation (4.1).

$$u(x,0) = \sum_{n=1}^{N} A_n \sin(2\pi f_n x), \tag{4.1}$$

where:

- A_n are amplitudes of random sign and magnitude from 0.5 to 2.0;
- $f_n \in \{0.5, 1.0, 1.5, 2.0, \ldots, 5.0\}$ are the sine wave frequencies;

A combination of such wave components and the resulting solution is shown for reference in Figure 4.2. Four cases are chosen that lead to the development of increasingly difficult solution patterns: multiple 'shock' fronts, both stationary and travelling in space, and of increased velocity magnitude, shown in Figure 4.3 and labelled $2-5$. Initial conditions 2 and 3 are combinations of randomly weighted sin curves of random frequency; with 2 being made up of two high frequency curves and 3 being a combination of three lower frequency curves (Equations 4.2 and 4.3). Initial conditions 4 and 5 are given by Equations 4.4 and 4.5, strictly increasing the frequency and the amplitude respectively.

$$u(x,0) = 1.02\sin(4\pi x) - 0.21\sin(10\pi x), \tag{4.2}$$

$$u(x,0) = 0.62\sin(\pi x) + 1.81\sin(2\pi x) - 1.02\sin(3\pi x) \tag{4.3}$$

$$u(x,0) = \sin(2\pi x) \tag{4.4}$$

$$u(x,0) = 1.5\sin(\pi x) \tag{4.5}$$

The next variation of our benchmark was generated by adjusting the value of ν from the original $\nu = \frac{1}{100\pi}$ to $\nu = 0.01$ and $\nu = 0.001$. As the magnitude of the diffusion term decreases, the steepness of the solution gradient increases, raising the problem complexity, and vice versa. This can be seen in the solution visualisations shown in Figure 4.1.

As for the second PDE, the Allen-Cahn equation is chosen due to its complex transient behaviours which are known to be challenging to capture computationally. It has also been used in similar research on PINNs such as Anagnostopoulos et al. [2023] and Penwarden et al. [2023]. In detail, the problem is defined as follows:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + 5(u - u^3), \quad x \in [-1, 1], \quad t \in [0, 1] \tag{4.6}$$

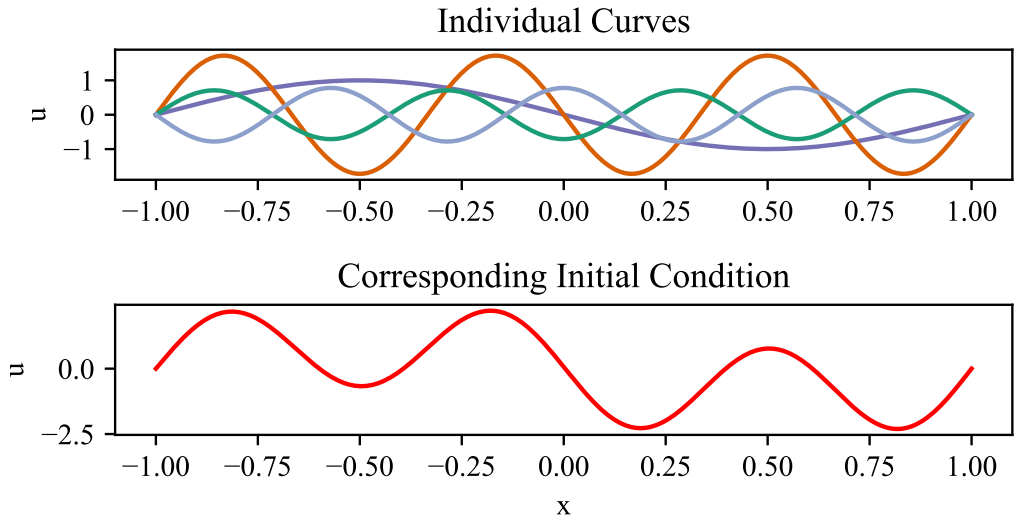

Figure 4.2: Example initial condition generation: via the sum of 4 curves with randomly selected amplitudes and frequencies (above) to create a more complex initial condition (below) that still satisfies the requirements for hard boundary condition enforcement.

We take D = 0.001 and use the initial condition

$$u(x, 0) = x^2 \cos(\pi x) \ ,$$

and boundary conditions

$$u(-1, t) = u(1, t) = -1 \ .$$

A visualisation of the ground truth solution for these initial and boundary conditions is provided in Figure 4.4.

4.2.2 Construction and validation of ground truth solutions

Finite Difference Methods (FDM) were used as the main approach for generating ground truth solutions. For the default initial condition, the ground truth solution had already been obtained and was already available from the work in the previous chapter. For the alternative cases described in the previous subsection 4.2.1, as the boundary conditions were still Dirichlet, and the initial conditions were simple trigonometric functions, the same FDM method was applied with ease.

For the Burgers' Equation, the only changes were to the initial condition and the value of the parameter ν in the PDE. Again, convergence was checked by increasing the resolution of the grid (Fig 4.5) and, upon obtaining a ground truth solution with enough accuracy, the grid was restricted to a smaller size.

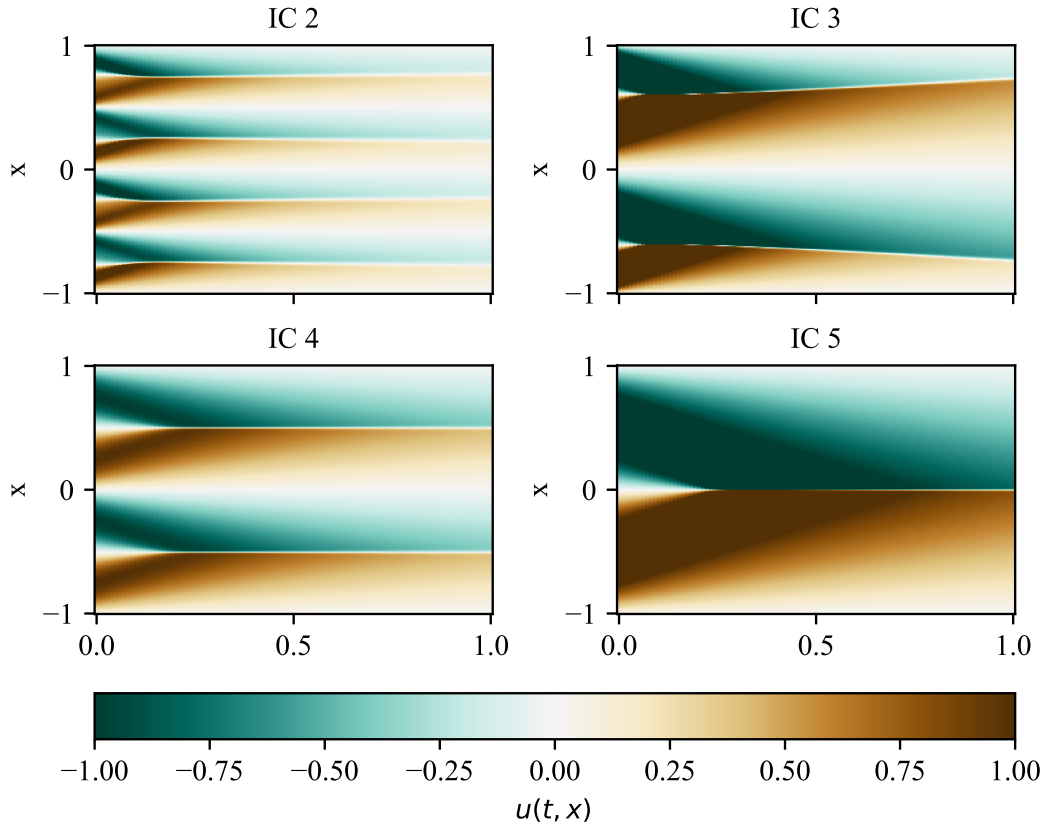

Figure 4.3: Reference solutions to Burgers' Equation with alternative Initial Conditions, defined by Equations 4.2-4.5.

For the Allen-Cahn benchmark, the open-source code provided by the authors of Wu et al. [2023] on GitHub contained both a solution file for our domain and parameters, and code that allowed us to generate our own solution files.

4.2.3 Implementation of guiding indicators

The main contribution of our work in this chapter uses solution estimates of velocity gradients, and the derivatives of residuals with respect to the spatial and temporal dimensions. Using solution gradients to identify the 'shock' region closely parallels adaptive mesh refinement seen in computational fluid dynamics applications ANSYS Inc. [2024]. In these approaches, a low resolution simulation is first run on a coarse mesh, and the mesh is subsequently refined in areas with large gradients, under the assumption that these regions correspond to where the dominant physical phenomena are located, which are necessary to capture accurately.

In our PINNs approach, we use a probability distribution function to guide the placement of points, as described in section 3.2.4. Up to this point, this probability distribution has been selected to bias collocation points towards areas of predicted high residuals. In order to vary the guiding information, all we thus need to do is efficiently compute the alternative guiding quantity at this step and use it to

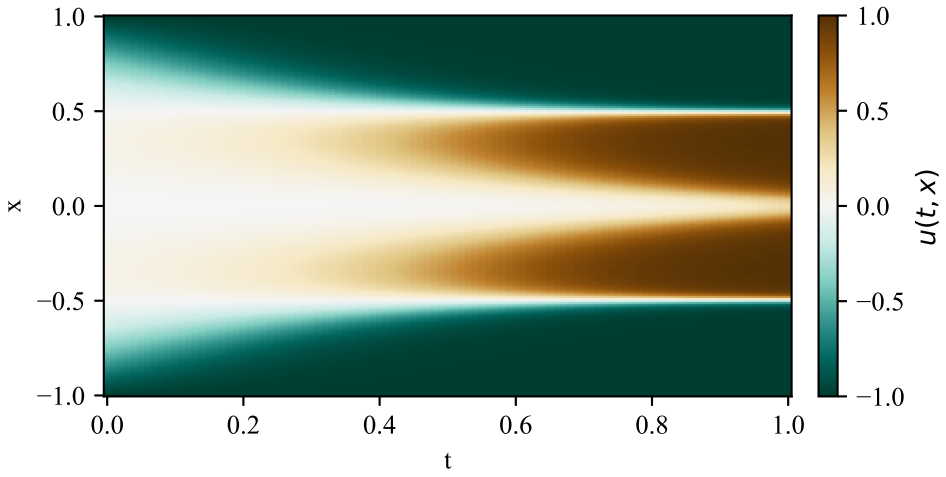

Figure 4.4: Reference solution to the Allen-Cahn equation.

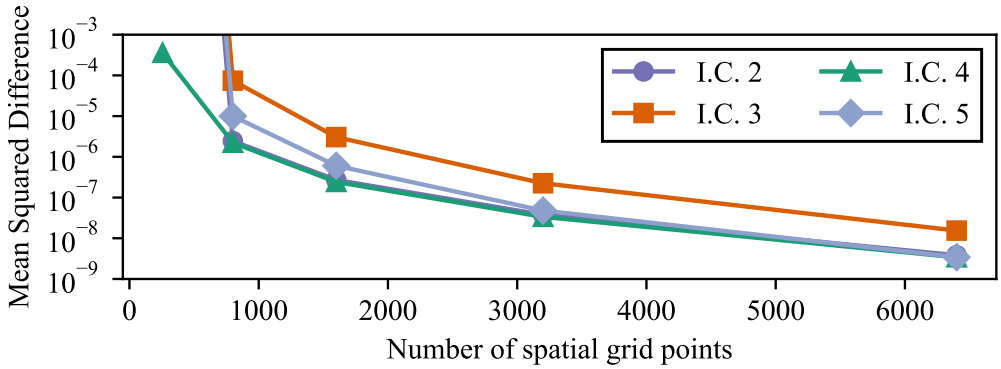

Figure 4.5: Grid sensitivity study - mean squared difference (MSD) computed between FDM solutions on coarser grids and a reference solution computed on the finest grid tested (12800×3201). This is plotted against the spatial resolution of the grid, which together with the temporal resolution is successively doubled.

calculate our probability distribution function instead of the residual, which is still used to minimise the loss function separately.

The information source used previously, the residual, consists of the terms of the PDE at each point. For the example of Burgers' Equation, the derivatives of velocity with respect to space, u_x and u_{xx}, and with respect to time, u_t, are obtained at every collocation point. This is done by performing a built-in operation in the DeepXDE library that computes the Jacobian matrix given an output tensor of the velocity and an input tensor of the spatial and temporal dimensions. From this matrix, the specified entry $J[i, j] = du_i/dx_j$ is returned, where the index i indicates the output component to support vector outputs (in this scalar case, $i = 0$ always), and the index j represents the dimensions with respect to which we differentiate (x, t). The second-order derivatives can be similarly obtained from computing a Hessian matrix H as $H[i, j] = d^2u/dx_i dx_j$, where this time both indices i and j represent the

input dimensions from which we wish to take derivatives. Returning to the Burgers' Equation example above, we make two Jacobian calls for $j = 0$ and for $j = 1$, and a Hessian call for $i = 1$, $j = 1$, returning vectors containing the derivatives u_x, u_t, and u_{xx} at every collocation point. By adding and multiplying these terms as defined by the PDE, the residual across all collocation points is thus obtained. To obtain instead information such as the curvature of the velocity field with respect to x and t for use as a guiding indicator, the Hessian function can again be called with the appropriate indices, which returns a vector that provides the curvature across all points.

The DeepXDE Jacobian and Hessian operations are not limited to taking velocity vectors as an input. By using the vector of residuals at collocation point set as the input tensor then, we can obtain the gradients of residual with respect to the spatial and temporal dimensions. This information captures where the physical laws aren't being satisfied (where the residual is high), with special attention to areas where sharp gradients in the solution are present. Cost wise, this evaluation is efficient and proportionately cheap in comparison to the cost of optimisation.

For a visual representation of how these different sources of information influence sampling, collocation point density heatmaps summarising the behaviour throughout training are presented in Section A.1 of Appendix A.

A number of conceptual considerations arise when replacing the residual with alternative guiding quantities:

- Directly substituting higher-order derivative information without normalisation or scaling may produce values of much larger magnitude, which could then dominate the probability distribution. This could cause the probability distribution function (PDF) to overly concentrate and cluster points, failing to identify broader regions of interest or secondary areas of interest. However, this effect could also be mitigated by adjusting hyperparameters k and c - the focus of the following subsection.

- A second consideration involves the treatment of multiple first–order derivatives. Because the problem is defined over two dimensions, x and t, the use of first derivatives as guiding information results in two distinct components; in higher dimensions this number would increase accordingly, making the choice of combination strategy increasingly important. For each collocation point it must therefore be determined how these contributions should be combined. Possible approaches include summation, averaging, or taking the maximum,

and the influence of these on model performance is examined as part of the study.

- Other important design choices also arise regarding implementation. When utilising indicators that rely on a solution estimate, carrying out a sufficient number of training steps prior to the first resample becomes significant, as the solution must be sufficiently converged to provide a reliable initial estimate of the location of high gradient regions and other areas of physical interest. In practice, this was found to be primarily a question of selecting an appropriate delay before the first resampling stage, rather than a fundamental limitation of the approach.

- A final consideration is whether multiple guiding signals can be employed sequentially during model training, for example by using residual-based guidance initially and then switching to gradient-based estimates after a specified number of resamples. This sequential strategy was straightforward to implement and, along with the other possible approaches described above, is listed in table 4.1. All of these methods are evaluated in the following subsection, where their performance using different hyper-parameter settings is evaluated.

Label	**Description / Definition**
Fixed, R	Fixed point baseline, with random initial sampling
Fixed, H	Fixed point baseline, with Hammersley initial sampling
PDE, R	Residual with random initial sampling, as in Wu et al. [2023]
PDE, H	Residual with Hammersley initial sampling
U_{xt}	Mixed second derivative: $\dfrac{\partial^2 u}{\partial x\,\partial t}$
PDE_{xt}	Residual mixed second derivative: $\partial^2(PDE)/\partial x\,\partial t$
$\mathrm{Avg}(U_x, U_t)$	Arithmetic mean of first derivatives: $\frac{1}{2}\big(u_x + u_t\big)$
$\mathrm{Max}(U_x, U_t)$	Maximum of first derivatives: $\max\big(u_x,\, u_t\big)$
$\mathrm{RMS}(U_x, U_t)$	Root-mean-square of first derivatives: $\sqrt{(u_x^2 + u_t^2)}$
PDE - U_{xt}	Sequential use of residual and Uxt
PDE – $\mathrm{Avg}(U_x, U_t)$	Sequential use of residual and $\mathrm{Avg}(U_x, U_t)$
PDE – $\mathrm{Max}(U_x, U_t)$	Sequential use of residual and $\mathrm{Max}(U_x, U_t)$
PDE – $\mathrm{RMS}(U_x, U_t)$	Sequential use of residual and $\mathrm{RMS}(U_x, U_t)$

Table 4.1: Notation for all guiding indicators tested and indicator variant. The term "Sequential" indicates that the first method is applied to the initial 50% of resamples, and the second method to the remaining 50%.

4.2.4 Hyperparameter Tests

A key component of the adaptive resampling framework used in this work is the construction of a probability distribution function (PDF) that determines the se-

lection of new collocation points. Following this approach, drawn from Wu et al. [2023], the PDF is formulated with two tunable hyperparameters k and c in ((4.7)), with recommendation of 1,1 for residual resampling.

$$P(\mathcal{X}) = \frac{Y(\mathcal{X})^k}{\overline{Y(\mathcal{X})^k}} + c \tag{4.7}$$

In this chapter, $Y(\mathcal{X})$ denotes the value of the chosen indicator at point $\mathcal{X}$. Whilst $Y(\mathcal{X})$ corresponds to the PDE residual in the original formulation of Wu et al. [2023], here it is generalised to represent any guiding indicator in use. As an exponent, the parameter k controls how sharply the probability of selecting points scales with regions with high values of $Y(\mathcal{X})$, whether residual or alternative indicators. For the default $k = 1$, the probability distribution $P(\mathcal{X})$ scales linearly, whereas values of $k < 1$ can be useful to reduce the dominance of sharp peaks in $Y(\mathcal{X})$, as may occur when using gradient-based indicators. On the other hand, the constant c introduces a baseline probability to all points, adding some uniform noise to the sampling process. This helps ensure that the entire domain can still be represented after the resampling, outside the regions of high $Y(\mathcal{X})$.

The appropriate selection of these hyperparameters is found to play a significant role in the overall performance of the adaptive strategy, and the optimal values may vary depending on the specific guiding information used. While Wu et al. [2023] provide recommendations for typical values of $k = 1$, $c = 1$, these are based on residual-based resampling. Accordingly, an investigation into the effect of these parameters is conducted for the alternative indicators considered in this work. The results of this investigation, summarised in Figure 4.6, examine 4 sets of hyperparameters together with the different method variants described in section 4.2.3. These results led to the selection of values $k = 1$, $c = 1$ for most methods. In addition to those methods, the gradient of the PDE residual with respect to the input dimensions, labelled PDE_{xt}, was also tested, for which the choice $k = 0.5$, $c = 1$ provides a significant improvement in accuracy across L (Figure 4.7). Possibly, this could be a result of the reduction of the exponent k mitigating the the influence of the large values produced by higher-order derivatives on the probability distribution.

In addition to selecting suitable fixed values for the parameters k and c, more sophisticated strategies were also briefly explored. Specifically, we considered dynamic sampling schemes that would eliminate the need to manually choose c by decreasing it throughout training. In the approach tested here, c was reduced linearly from an initial value of 1 at the first resampling step to 0 at the final resample. This would

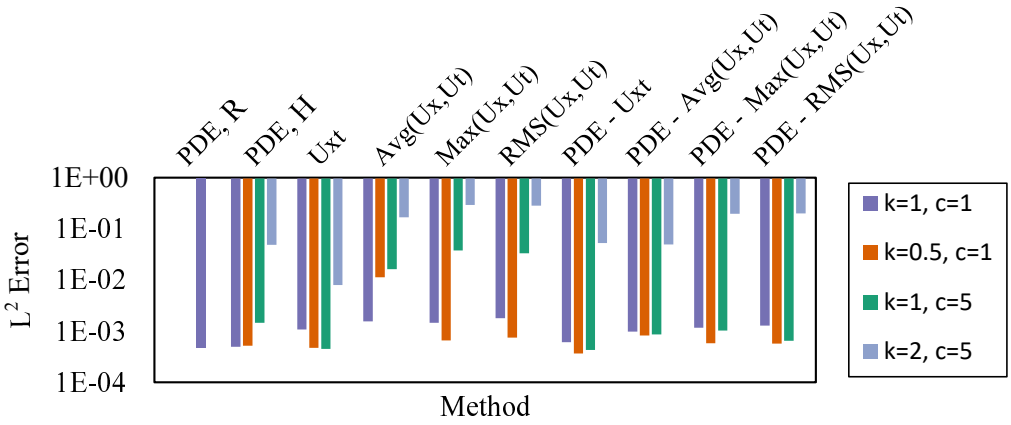

Figure 4.6: Graph showing error using 2000 collocation points, 100 resamples, for all methods listed in table 4.1, including the different implementations of first-order derivatives and the sequential methods (which switch the guiding indicator after the first 50 resamples). Within each case, bars represent the error of the different hyperparameter sets tested.

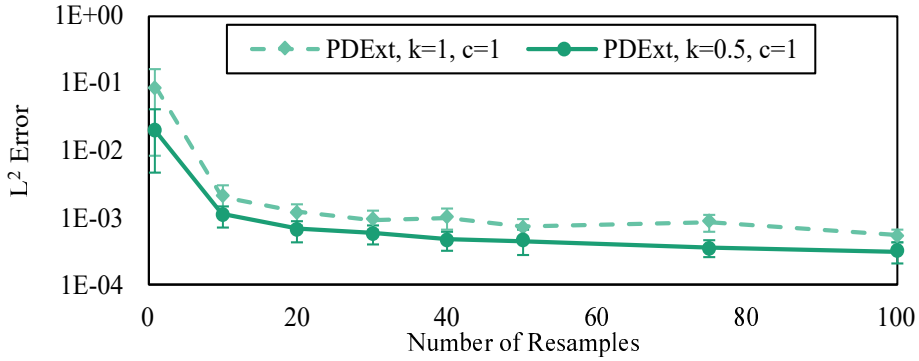

Figure 4.7: Hyperparameter study for the derivatives of the residual information source, PDE_{xt}, showing error against number of resamples L for $N = 2000$ and a network depth $D = 3$. Error bars represent ± 1 standard deviation (SD).

allow the sampling distribution to transition gradually from a set of adaptive points with noise to a fully indicator-driven set of points, analogous in spirit to cosine annealing strategies used in learning rate scheduler methods.

While these approaches offered flexibility and illustrated how resampling strategies could be adjusted over the course of training, they ultimately provided little improvement over the static k–c formulation. As they failed to remove the need for manual tuning altogether (because of hyperparameter k), are not considered further in this work.

4.2.5 Number of repeats

Due to the non-deterministic nature of the training process, repeated runs are required to ensure the computed average error is reflective of a given method's performance, and not influenced by outliers. While average errors are shown in the

plots presented throughout the chapter, the errors from individual runs are also considered. This allows for the identification of runs in which the model may have failed to train successfully, due to either resource limitations or problem complexity. In cases where such failures occurred and have a noticeable effect on lowering the average error of a given method, this is explicitly noted in the relevant discussion. To ensure that a sufficient number of repeats is used, the variation in the average error of the method with increasing number of repeats was examined. The results of this analysis are shown in Figure 4.8.

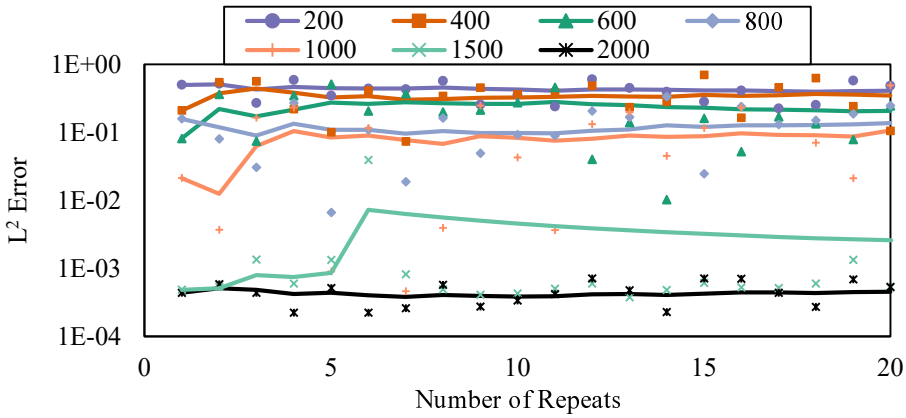

Figure 4.8: Convergence plot showing the L_2 error against the number of repeated runs, for different numbers of collocation points N. The dots indicate the individual errors for each run, while the lines show the cumulative average. The PDE, H method (residual-based with initial Hammersley sampling) was used, with $L = 100$.

We can observe that at extremes of N, runs are consistently low or high in error, with cases in between showing a mix of both good and bad runs. Cases like $N = 1000$, which include one run with an error of 10^{-3} but mostly performing around 10^{-1}, exhibit a consistent average error by the end of the range shown. The case of $N = 1500$, which mostly performs similarly to, or just slightly worse than, $N = 2000$, contains a particularly bad outlier that impacts the average. However, this is smoothed out by the end of the range. A value of 10 repeats was chosen, which seemed sufficient to account for variability, while remaining computationally efficient, as each run could be executed independently on a single CPU core with minimal overhead.

Finally, it is worth noting that several broader design choices arise in the construction of PINNs, beyond the considerations examined previously in this section. While some targeted checks have been carried out to ensure that the chosen settings are reasonable, an exhaustive exploration of architectural and PDF hyperparameters was beyond the scope of this study. Since the aim of this chapter is to assess the

robustness and comparative performance of different adaptive resampling strategies introduced, rather than to optimise a model to maximise overall predictive accuracy, a deliberately simple implementation was favoured (consisting of a fully connected network with manually selected training settings). This implementation thus helps isolate the effect of the various adaptive resampling methods tested, providing a better basis for comparison.

4.3 Results

This chapter evaluates the extent to which different sources of guidance in adaptive resampling strategies can improve the efficiency of PINNs. The central question is whether these techniques can yield comparable levels of accuracy to PINNs without adaptive resampling, while requiring fewer training points, thereby reducing computational cost.

To assess this, two primary metrics are considered. First, we examine how solution accuracy varies with the number of training points used. The number of points serves as a problem-agnostic measure of computational effort and allows for comparisons that may generalize across different problem settings. Second, we analyse the convergence behaviour of the training process - specifically, the number of training steps required for the solution to stabilize. For adaptive resampling methods, we approximate this by tracking the number of resampling events, each of which entails an additional round of training.

Finally, we also investigate the overall computational cost as the number of training points, N increases. This analysis helps determine whether the overhead introduced by resampling grows disproportionately with N, and whether this growth is consistent across different scenarios.

First, we look at the main comparison between the different guiding quantities for the default 1D Burgers' Equation benchmark. This is followed by all the variations to the benchmark discussed in Section 4.2.1 to check for the generalisability of our observations.

4.3.1 Burgers' Equation Results

Main case

For this initial test problem, six different solution strategies were applied using $N = 2000$ collocation points. In each case, the training regime was: 15,000 initial steps of ADAM optimization, followed by 1,000 steps of L-BFGS, with resampling

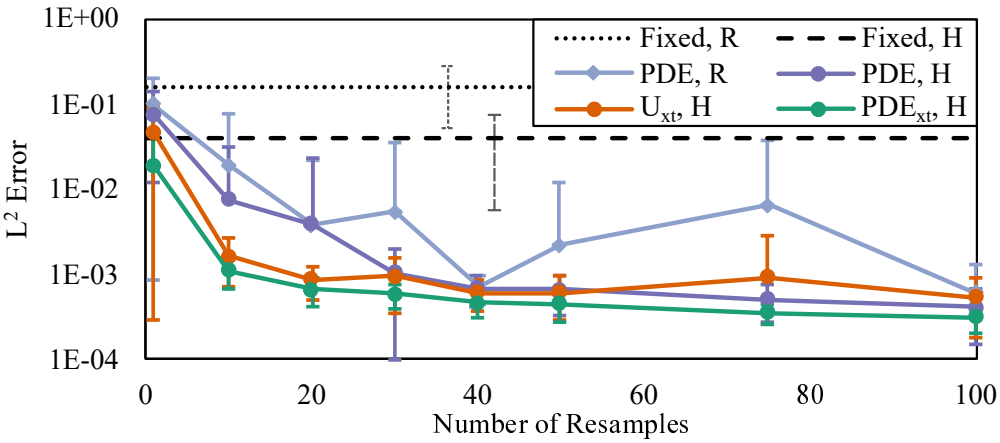

Figure 4.9: Error against number of resamples L using non-adaptive ('Fixed') and adaptive methods: PDE residual ('PDE'), second derivative of u (U_{xt}), and second derivative of PDE residual ('PDE_{xt}); using different initial distributions: uniform random ('R') and quasirandom ('H') initial sampling. Error bars represent ±1 SD, with the dashed error bars representing the SD of the two fixed methods.

conducted every 2,000 steps (1,000 ADAM steps followed by 1,000 L-BFGS steps), a regime consistent with Wu et al. [2023]. The initial Adam steps allow the model to converge to an initial coarse solution, with 15,000 being enough steps to ensure the initial loss has begun to plateau, at which point switching to the L-BFGS optimiser further decreases the loss. After every resample, the loss initially re-increases, and we found 2,000 steps per resample were enough to offset this and decrease the loss from its previous lowest point. The error is plotted against the number of resamples taken for each of the strategies and is shown in Figure 4.9. This graph includes a baseline of two constant point sets, one a uniform random distribution and another a distribution based upon the Hammersley sampling algorithm.

The initial distribution is seen to have an effect in both the fixed methods and residual-based resampling (despite the latter only using it for the first 15,000 Adam iterations), with the use of the Hammersley sampling algorithm resulting in consistently lower error. The local residual method with random initialisation (PDE, R) exhibits the least robustness and greatest variability, which can be seen from the fact increasing the number of resamples does not consistently lower the average error between runs. This higher average error can be attributed to a high proportion of individual runs that does not converge to good global minima and have much higher errors. As for the other adaptive methods, they exhibit less variability and see error decreasing below 0.1% above 30 resamples. The two derivative-based guiding indicators, U_{xt} and PDE_{xt}, seem to also slightly outperform PDE, H when fewer than 30 resamples are employed, for instance with 10 resamples achieving errors of

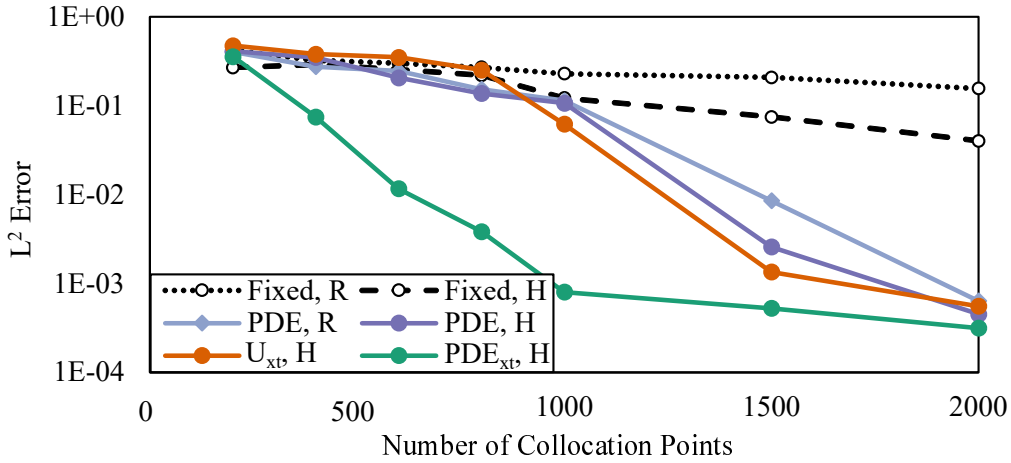

Figure 4.10: Error against number of collocation points N with $L = 100$, using non-adaptive ('Fixed') and adaptive methods: PDE residual ('PDE'), second derivative of u (U_{xt}), and second derivative of PDE residual ('PDE_{xt}); using different initial distributions: uniform random ('R') and quasirandom ('H') initial sampling.

0.109% and 0.175%, respectively, compared with 0.75% error for the PDE, H.

For the same strategies, the error achieved varying the number of collocation points is next presented, in Figure 4.10. These were obtained using 100 rounds of resampling, equivalent to 215,000 steps, the amount of optimisation steps used for fixed set methods.

In Figure 4.10, the number of points required to achieve a specified level of accuracy is seen to vary across the methods considered. The most accurate method from the test presented in Figure 4.9, PDE_{xt}, sees the error plateau within the range of 0.01% to 0.1% around $N = 1,000$, an accuracy that is only reached by the other adaptive methods around the default $N = 2,000$ points. The error of the fixed Hammersley set also decreases with number of collocation points, but only reaches $\approx 9\%$ at $N = 2,000$.

To complement the accuracy comparisons, Figure 4.11 reports the associated computational costs of the main methods considered. The time shown is total wall clock time, as run on single CPU cores (Intel Xeon Gold 6138 CPUs, of up to 2GHz clock rate). The cost scales approximately linearly with the number of resamples over the range tested, with little difference observed under 40 resamples. However, the cost of the PDE_{xt} indicator method increases slightly more rapidly, becoming the most expensive method at 100 resamples (equivalent to 216,000 combined Adam and L-BFGS optimisation steps).

To ensure their convergence, no-resampling fixed methods were run for an equivalent number of total steps using an L-BFGS optimiser, requiring approximately

10 hours on average. However, the loss curves for these methods plateaued after around 30,000 steps, depending on the problem and initial condition, indicating that optimisation beyond that number of steps yielded marginal improvements of the associated L_2 error. As a result, implementing early-stopping could significantly reduce the runtime of these baseline methods to under 1 hour.

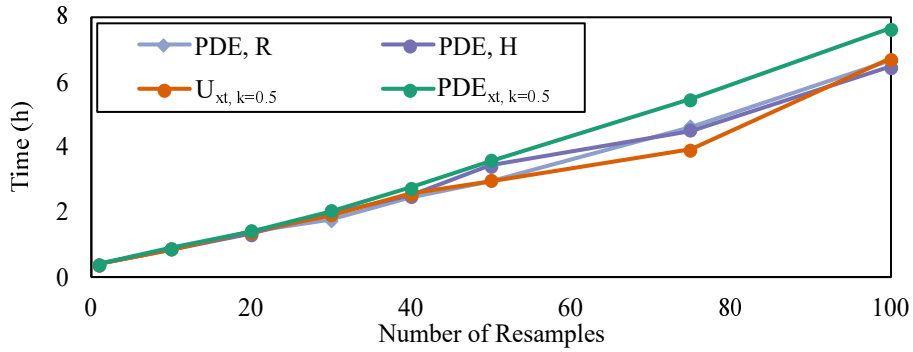

Figure 4.11: Cost (in hours) of resampling using different indicators, using a depth of 3 and $N = 2000$.

We can observe that the cost mostly scales quite linearly with number of resamples (and therefore optimisation steps). The cost of the double derivative of the residual (PDE_{xt}) can be seen to increase slightly faster with resamples than the other methods. By referencing back to Figure 4.9, what's important to note is that using the double derivative methods (U_{xt} and PDE_{xt}) allows the PINN to achieve errors under 1×10^{-3} in only 20 resamples (1.4h), compared to 40 resamples for residual based resampling (2.5h), and unlike the fixed methods at $N = 2000$.

Other initial conditions

To assess the generalisability of the findings from the default case, this section investigates whether the advantages observed for the PDE_{xt} and other adaptive methods persist when the shape and complexity of the initial state are varied. To do so, the error against L is presented in Figure 4.12 for the various initial conditions described previously in Section 4.2.2.

In three of the four initial conditions shown (2, 4 and 5), PDE_{xt} clearly outperforms the other methods (as it did for initial condition 1 in the previous section). Not only does it deliver a smaller error after the last resample (the maximum amount of training investigated) but the error curve lies beneath the others at each stage in the adaptive process.

This gap is most noticeable in the 5th initial condition, which corresponds to a case with a larger amplitude, thus leading to a faster and sharper 'shock'. In this

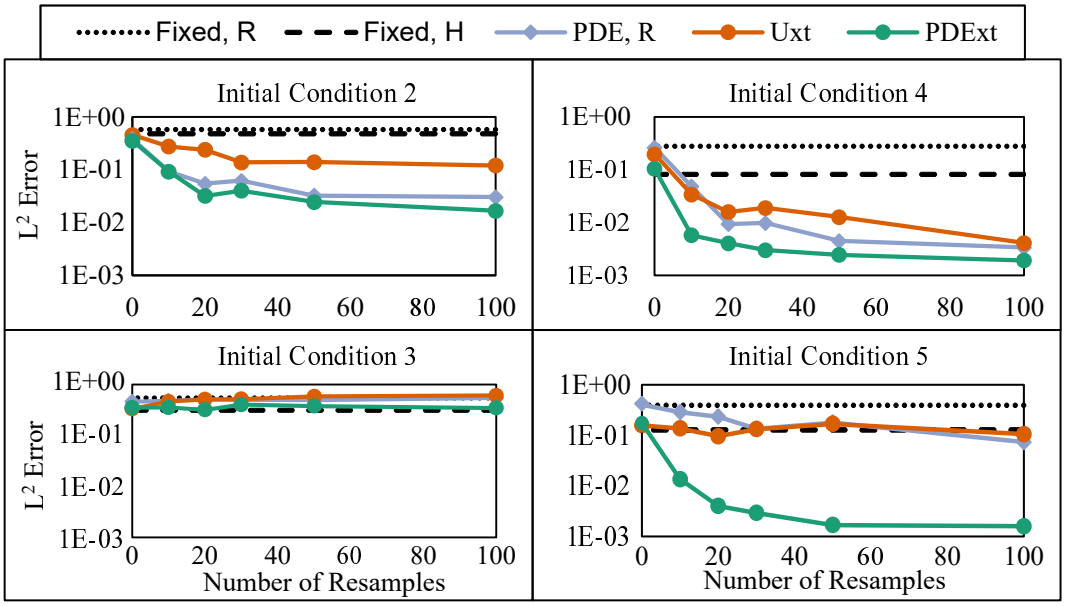

Figure 4.12: Error against number of resamples using different adaptive and non-adaptive resampling methods, across the four different initial conditions tested.

case it far outperforms the next best method.

Furthermore, in each case other than initial condition 3, the adaptive collocation point algorithms all performed significantly better than using fixed collocation points. Unsurprisingly, of the two fixed point approaches Hammersley always outperforms the uniform random distribution.

The exception is initial condition 3 where all methods perform poorly, with none achieving an error below 9%. This is the most complex initial condition considered, containing the the highest values of u and therefore the sharpest 'shocks'. From the results, it thus appears that no method can compensate for the lack of sufficient collocation points (or possibly a more expressive network).

4.3.2 Viscosity parameter, ν

For the following examples, the value of the ν parameter in the PDE is adjusted to test the generalisability of the adaptive methods when applied to problems of varying difficulty, while still retaining the same overall structure. This serves as an intermediate step before investigating more substantial changes to the PDE itself. The parameter ν affects the magnitude of the diffusion term u_{xx}: lower values increase shock sharpness and thereby increase problem complexity, while higher values smooth the solution and reduce computational difficulty. Values of ν equal to 0.01 and 0.001 are used, approximately representing a multiplication and division by 3 relative to the default $\nu = \frac{1}{100\pi}$ employed previously. The error associated with different information sources used to control the adaptive resampling is compared

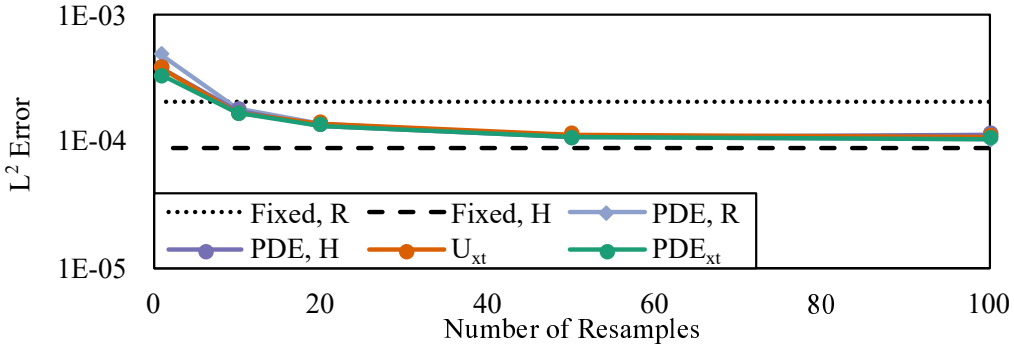

Figure 4.13: Error against number of resamples L for a viscosity $\nu = 0.01$

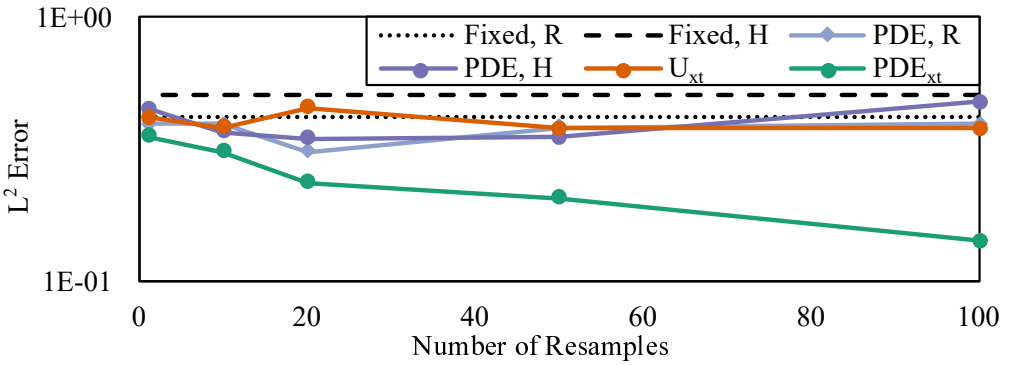

Figure 4.14: Error against number of resamples L for a viscosity $\nu = 0.001$

against that of fixed uniform methods by plotting error against number of resamples; see figures 4.13 and 4.14.

For the simpler case, $\nu = 0.01$ (Figure 4.13), all adaptive methods behave similarly, delivering a small error after relatively few resamples. Comparing the fixed methods, the Hammersley distribution is again consistently better than a random fixed distribution. Furthermore, in this specific example, it also outperforms the adaptive sampling methods. To assess the effect of using fewer collocation points in this simpler case, a test is carried out (table 4.2). Here, as the number of available points decreases, the impact of adaptive resampling is much more pronounced: the previously most accurate 'Fixed, H' non-adaptive method shows an increase in error of over 300%, whereas the residual second-derivative method 'PDE_{xt}' exhibits only a marginal increase in error of 19%, despite using half the collocation points.

N	**Fixed, R**	**Fixed, H**	**PDE, R**	**PDE, H**	**Uxt**	**PDExt**
1000	3.09E-03	3.93E-04	1.50E-04	1.87E-04	1.46E-04	1.26E-04
2000	2.02E-04	8.95E-05	1.02E-04	1.15E-04	1.09E-04	1.06E-04
Change	1429.9%	338.4%	46.8%	62.1%	33.9%	19.3%

Table 4.2: L_2 Error results of different adaptive and non-adaptive methods, $\nu = 0.01$, $L = 100$

For the more challenging case of $\nu = 0.001$ (Figure 4.14), all methods struggle to converge to an accurate solution using $N = 2000$ collocation points. The best approach, with an L_2 error of 14%, is again the adaptive method based upon PDE_{xt}. To assess the effect of increasing the number of collocation points in this more complex regime, results for N = 4,000 are shown in table 4.3. In this case, the reduction in error is uneven across methods, with the residual-based and residual-derivative approaches benefiting most, and the latter decreasing to 3.67% error. These results indicate that adaptive resampling not only reduces the computational effort required for simpler cases, but may also enable the solution of more challenging problems.

N	**Fixed, R**	**Fixed, H**	**PDE, R**	**PDE, H**	**Uxt**	**PDExt**
2000	4.209E-01	5.093E-01	3.925E-01	4.780E-01	3.797E-01	1.425E-01
4000	4.049E-01	3.836E-01	1.858E-01	1.996E-01	2.591E-01	3.660E-02
Change	-3.9%	-32.8%	-111.2%	-139.5%	-46.6%	-289.3%

Table 4.3: L_2 Error results of different adaptive and non-adaptive methods, $\nu = 0.001$, $L = 100$

4.3.3 Allen Cahn Results

In this subsection we consider the application of the adaptive resampling strategies on the Allen–Cahn equation to evaluate their effectiveness beyond the Burgers' equation. For this problem we again compute a high-resolution numerical solution for the ground truth and consider the L_2 error as the number of resamples is increased. Based on the observations of the previous subsections, we undertake these tests for three distinct numbers of collocation points: N = 500, 1000, 2000. The results for $N = 500$ are shown in Figure 4.15.

For this case, the adaptive methods are again consistently superior to using fixed collocation points. Similar to the case of Burgers' Equation with an increased viscosity parameter, $v = 0.01$, all the adaptive approaches perform comparably, with U_{xt} slightly behind. The very high error of the fixed collocation point methods suggests that for this low value of N there are so few points that a non-biased set of points cannot be used to train a PINN successfully. The adaptive methods of resampling, whilst not achieving as low an error as previously observed in Burgers' Equation, are still able to train the network and produce errors of less than 1% at $L = 100$. This is potentially due to the ability of the adaptive resampling to provide new points for learning with every resample.

Moving on to the cases with higher number of collocation points, for $N \geq 1000$,

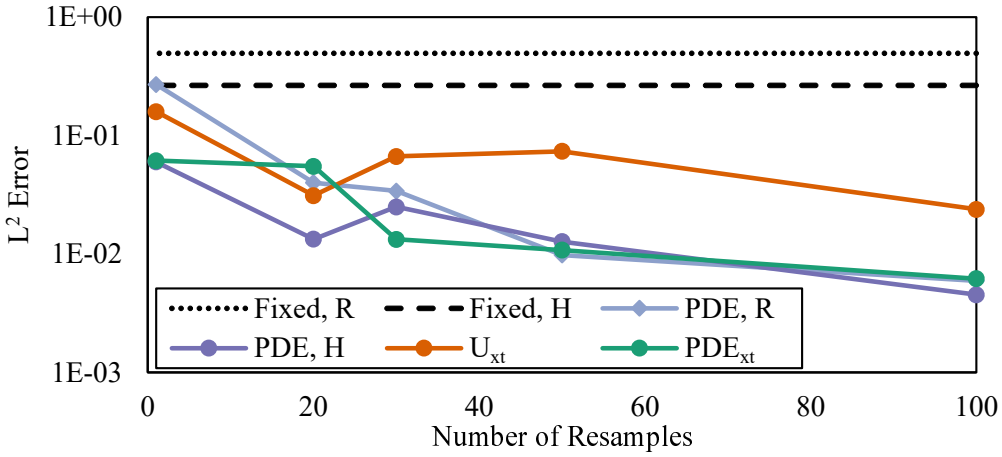

Figure 4.15: Error against number of resamples L, using $N = 500$, solving the Allen-Cahn PDE.

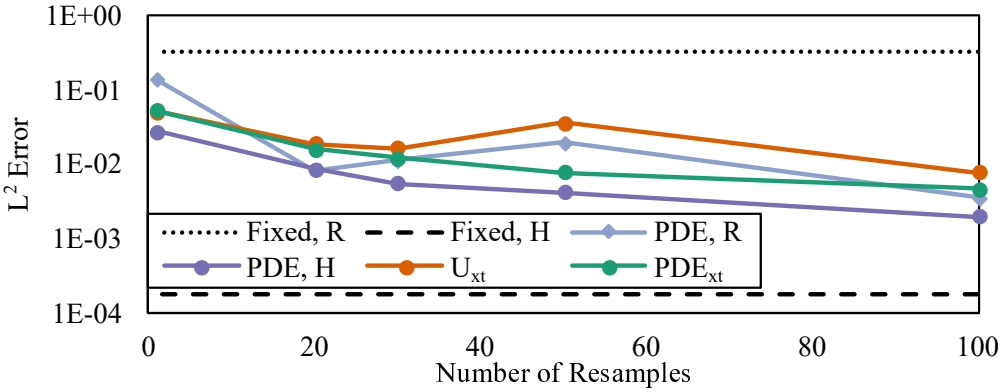

Figure 4.16: Error against number of resamples L, using $N = 1000$

the results are shown in figures 4.16 and 4.17. For $N \geq 1000$, the adaptive resampling methods do not show significant improvement over a non-adaptive quasirandom distribution of collocation points, and while a uniform random method still fails to train, the fixed method using a quasi-random distribution achieves the lowest error observed. For both $N = 1000$ and $N = 2000$, the increased number of points allows the domain to be sufficiently covered with an appropriate fixed distribution. As a result, redistributing points provides limited additional benefit, which may even hinder training due to the frequent optimiser resets, whereas the fixed method completes training using L-BFGS exclusively after the initial training phase with Adam.

For the Allen–Cahn equation, the influence and effectiveness of sampling strategies appear to vary substantially with the number of collocation points used. The less abrupt gradients in this solution may also reduce the impact of derivative-based resampling criteria. Nevertheless, for $N = 500$, the adaptive methods still clearly outperform fixed-point distributions, reinforcing the idea that comparable accuracy

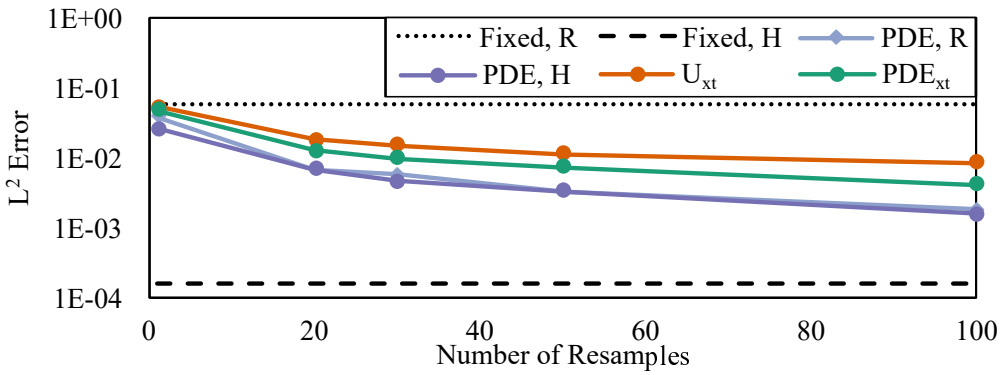

Figure 4.17: Error against number of resamples L, using $N = 2000$

can be achieved with fewer points when adaptive sampling is employed. Overall, these results suggest that once a sufficiently dense set of collocation points is used, the benefit of adaptive resampling disappear, whereas in regimes with fewer points (and therefore lower cost of loss evaluation) adaptive strategies can still provide clear gains in accuracy and efficiency.

4.4 Discussion

4.4.1 Overview

In this chapter, to better explore the problem of collocation point sampling in PINNs we have examined the two benchmark problems of Burgers' Equation and the Allen-Cahn Equation. We have used guiding indicators based on derivatives of the local residual and the solution estimate to guide the resampling process, and compared the results to other resampling methods and fixed random and pseudo-random distributions.

Overall, we have observed that using adaptive point distribution typically allows more accurate PINNs solutions to be obtained than using fixed point distributions for small numbers of collocation points. Using fewer collocation points is potentially important since the training cost of a PINN grows in proportion to this number (all other factors being equal). Furthermore, we have observed that defining the probability distribution for the collocation point locations based upon the mixed second derivative of the residual with respect to the independent variables typically outperforms, often significantly, using the residual alone, as in Wu et al. [2023].

For Burgers' equation with a smooth initial condition that leads to the formation of a single "shock", and a diffusion parameter such that this feature is sufficiently sharp, we obtained the smallest errors using this approach (Fig. 4.9). Equally importantly, we observed that we could reach such errors with significantly fewer

collocation points than for any of the other techniques that we compared against (Fig. 4.10). When considering other initial conditions and other diffusion parameters the relative performance of the adaptive approaches was seen to depend upon the complexity of the specific solution in each case (Fig. 4.12). Typically, there appears to be a value of N above which the use of a uniform (Hammersley) distribution of points is preferred - however this value can be relatively large for the more complex problems (and is relatively small when the solution is smoothly varying everywhere). When considering numbers of points below this critical value of N we invariably find that using the mixed derivative of the residual is the best strategy for defining the PDF for drawing the collocation point locations.

Consideration of a different PDE, the Allen-Cahn equation in this case in Figure 4.15, demonstrates similar behaviour. Specifically, for a given problem there is a value of N below which the adaptive sampling approach is preferable (and above which a uniform Hammersley distribution suffices). In the particular examples considered here, defining the PDF based upon the mixed derivative of the residual is still reliable and robust, however its improvement relative to using the residual alone is less clear.

4.4.2 Limitations and Next Steps

In terms of computational cost, the PINNs network we have considered is not competitive with standard discretization methods for solving individual forward problems. This is primarily due to the high computational cost of training with full gradient descent methods like L_BFGS. As for the cost of the resampling strategies we have discussed, these only contribute to a small overhead (estimated at 3% in Wu et al. [2023]) to the overall computational cost of PINNs training. While the cost of training a PINN compared to solving numerically is high, this study remains valuable for two reasons. First, as the cost stems principally from the optimisation algorithm, which scales with the number of collocation points N, methods such as our adaptive resampling that can reliably reduce N will directly reduce the training cost both now and as more efficient optimisation algorithms are developed. Second, PINNs have a high potential for tackling problems for which traditional numerical solvers may be difficult or impossible to apply, such as cases with unknowns or partial boundary information. In such cases, the PINN formulation is flexible, being able to incorporate soft boundary constraints or available data to make the problem well-posed.

Generally, appropriately training a PINN is a challenge Wang et al. [2023a],

specifically determining what constitutes an appropriate training regime. In the case with no data that we have considered, overfitting was not an issue we had to consider or observed, as the number of training points used was not limited by available data, but by the computational cost of adding collocation points. However, the choice of optimisation algorithm and training regime was still impactful. The use of faster stochastic gradient descent methods like ADAM helped initialise training but had a limited ceiling to accuracy, hence the need for the more expensive L_BFGS. Whilst we were not able to fully explore the impact of different training regimes within the scope of this chapter, efficiency could feasibly be improved by further optimising this via methods such as cosine annealing Subramanian et al. [2023]; fine-tuning architecture hyperparameters; or more recently looking at the optimisation algorithms themselves [Urbán et al., 2025].

Beyond the choice of optimisation strategy, other implementation-level tests were conducted to assess the sensitivity of our resampling methods to various design choices such as hyper-parameters, management of multiple information sources (such as for the various first order derivatives of the solution or residual), and the hybrid use of information sources, which we briefly considered by looking at sequential changes. We also developed and explored dynamic scheme for the parameter c by gradually decreasing it during training, smoothly transitioning from a noisy to a purely indicator-driven resampling. However, these implementation variations rapidly expanded the scope of investigation, making a comprehensive study impractical within this work. As such, we used $k = c = 1$ as a reasonable default in most cases, when changes to these values did not yield substantial performance improvements. The exception was the PDE_{xt} case, where using a less aggressive weighting by setting $k = 0.5$ proved more effective. This adjustment helps to flatten the probability distribution, preventing regions with very high $Y(\mathcal{X})$ from dominating the sampling, an issue that becomes more pronounced when computing higher-order derivatives over sharp gradients.

In summary, we have shown that a number of different strategies are viable for selecting the location of PINNs collocation points with resampling methods. In particular, we have shown that resampling can be guided not only by the local residuals, but also by the spatial and temporal derivatives of the residuals and even of the solution estimates themselves. Furthermore, we have observed that a number of different factors impact the accuracy of the PINN method, making it difficult to analyse any single aspect (such as collocation point selection) in isolation. Whilst we believe that our contribution begins to shed some light on the interplay between the

location of the collocation points and the ability of a PINN to learn the PDE solution it is clear that further research is desirable in order to be able to propose robust implementations that support reliable training or provide accuracy guarantees.

Nevertheless, the analysis presented in this chapter highlights the ability of adaptive resampling to reduce the number of collocation points required for training a given problem (lowering the computational cost of evaluating the loss and optimisation), and the ability of certain guiding indicators to reduce the number of resamples (and thus training iterations) needed to achieve minimum error. In this chapter, these benefits have been demonstrated across a broad range of problems by varying initial conditions, PDE parameters, and testing two distinct PDEs. However, these investigations were limited to single PDEs in one-dimensional unsteady settings. This raises the question of whether the presented methods, along with the observed advantages, can generalise to higher-dimensional problems and to systems of PDEs. Addressing this question is the focus of the following chapter, which examines the practical challenges of extending adaptive resampling to a two-dimensional coupled system, evaluating the performance of both adaptive resampling itself and of alternative guiding indicators.

Chapter 5

Generalisability to Higher Dimensions

In this chapter we extend the adaptive resampling methods explored in Chapter 4 to a higher dimension problem. We thus investigate whether as dimensionality increases the benefits to performance (that allowed less points to be needed for a given accuracy) remain and so help the scalability of PINNs into higher dimensional problems.

The first section introduces our aims for the work carried out in this chapter and its role within the broader literature. Next, the methods are covered, from the 2D problem to the updates to the method presented so far in Chapters 3 and 4. Lastly, we show the results obtained and discuss the benefits and issues of the developed method.

5.1 Introduction

In our previous chapters we have explored how collocation point sampling in PINNs can be a valuable and effective component of the training process. Contributing to this, adaptive resampling can be implemented to automatically distribute points in areas that help maximise the effectiveness of the training. This has benefits to both accuracy, as lower errors can be achieved, and cost, as less points may be necessary to achieve a given accuracy. This was followed up in Chapter 4 by developing and examining the efficacy of different indicators used to guide the adaptive resampling process compared to using a fixed distribution of points. Our findings suggest that adaptively refining the collocation point distribution based on geometric properties of the estimated solution can lead to more accurate PINN solutions with fewer collocation points, striking a balance between computational efficiency and accuracy.

More generally, the potential is that adaptive resampling enables PINNs to scale more effectively to complex PDEs or systems of PDEs, making them more competitive as forward solvers in practical applications. This chapter, therefore, builds on previous work by extending the investigation of the same nonlinear convection-diffusion problem to a two-dimensional, unsteady domain. By applying adaptive point sampling strategies informed by solution geometry, we aim to better understand how different sampling methods interact with problem complexity and computational cost, thereby offering insights into optimising PINN performance in more challenging settings.

5.2 Problem Description and Methodology

5.2.1 2D Burgers' System of Equations

In two spatial dimensions, we replace the single equation seen in the previous chapters by the following system of equations, to obtain horizontal and vertical components of velocity separately, u and v respectively:

$$u_t + uu_x + vu_y - \nu\left(u_{xx} + u_{yy}\right) = 0, \quad x, y, t \in [0, 1], \tag{5.1}$$

$$v_t + uv_x + vv_y - \nu\left(v_{xx} + v_{yy}\right) = 0, \quad x, y, t \in [0, 1]. \tag{5.2}$$

As a default we again select $\nu = \frac{1}{100\pi}$, Dirichlet $u = v = 0$ boundary conditions at every wall and the initial conditions:

$$u(x, y, 0) = \sin(2\pi x)\sin(2\pi y); \quad v(x, y, 0) = \sin(\pi x)\sin(\pi y). \tag{5.3}$$

A representative solution to this system is shown in Figure 5.1. The main feature is still a sharp change in velocity in the x dimension, u, which travels upwards in time as it dissipates. There is also a sharp change in the other component, v, near the upper boundary (a boundary layer).

In principle, Burgers' Equation could be generalised to a single PDE in 2D or to a pair of coupled 1D PDEs. Here we adopt the more general approach of a coupled system of of two 2D PDEs for u and v. This choice provides a more demanding benchmark, but allows us to assess both the impact on scalability of increased domain dimensionality, and the impact of incorporating multiple PDEs into the loss function and into the adaptive resampling process.

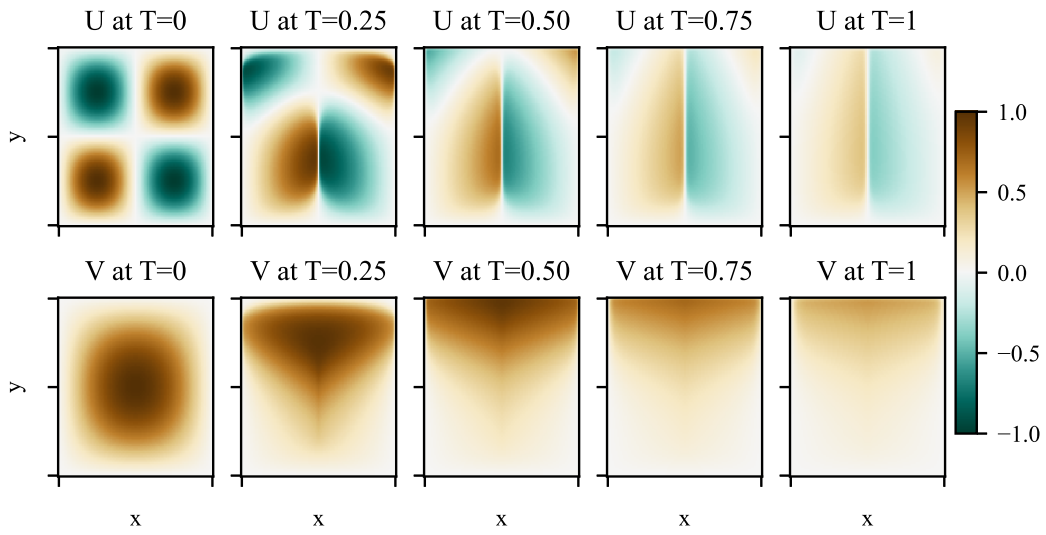

Figure 5.1: Reference solution of the 2D Burgers' system of equations. Above, component u; below, component v. Note the formation of the internal "shock" in the u component and the boundary layer in the v component.

5.2.2 Ground truth generation and validation

The ground truth data was generated using a MATLAB implementation developed for prior work by Mathias et al. [2022] on the 2D Burgers' Equation. The numerical scheme employed a sixth-order accurate spatial discretisation and a fourth-order Runge-Kutta method for time integration. Like in the previous chapter, the accuracy of this ground truth solution was measured by increasing the resolution and noting the difference, measured as mean absolute error (MAE). This MATLAB implementation was also used to produce a ground truth solution for a higher-ν case, which served as a simpler benchmark case, for which the same grid density was used.

5.2.3 Resampling in 2D

The methodology described in this chapter builds directly upon that of the one-dimensional case previously described, with some updates mentioned in this section. First, the loss function now simultaneously minimises the residuals of the two separate PDE losses (see equation 5.4) - where each collocation point is used to compute both PDE residual components. Weighting was not used for these; their contributions were simply added together. Similarly, all resampling indicators (residual and solution estimates) now have two components.

$$\begin{aligned} \mathcal{L}_u &= u_t + uu_x + vu_y - \nu(u_{xx} + u_{yy}) \\ \mathcal{L}_v &= v_t + uv_x + vv_y - \nu(v_{xx} + v_{yy}) \end{aligned} \tag{5.4}$$

The notation used to denote the different fixed and adaptive resampling methods

in this chapter follows the same convention as in previous chapters. Fixed-point baselines are denoted as 'Fixed, R' for uniformly random distributions, and 'Fixed, H' for quasi-random Hammersley distributions. Adaptive approaches are labelled according to the quantity used to guide resampling, with a subscript indicating the order of the derivatives taken with respect to spatial and temporal dimensions. 'PDE' refers to the sum of the two PDE residuals, while PDE_{1st} denotes resampling based on the sum of all the first derivatives of the PDE residuals, i.e. $\frac{\partial f(\mathbf{X})}{\partial x} + \frac{\partial f(\mathbf{X})}{\partial y} + \frac{\partial f(\mathbf{X})}{\partial t}$. U_{3rd} refers to resampling based on third-order mixed partial derivatives of the solution estimate, where the two components of velocity $\partial^3 u/\partial x \partial y \partial t$ and $\partial^3 v/\partial x \partial y \partial t$ are summed together.

Initial experiments used the same neural network architecture (namely a depth of 3 layers with 64 neurons each) as the one-dimensional implementation, similarly using tanh activation functions and a hybrid Adam/L-BFGS optimisation method. Later tests explored deeper network architectures, with depth increased to 5 and 8 layers. Modifications were also made to the resampling procedure itself for these later tests, focused on the training strategy. Namely, testing was done on reducing the number of resampling events while increasing the number of training steps performed on each set of points, motivated by observations that the 1D resampling strategy did not transfer effectively. The problems with the approach from 1D are shown in Section 5.3.1, with the testing on number of resamples being the focus of Section 5.3.2.

As part of the analysis in this chapter, loss history plots are used more prominently as a post-processing tool to assess training behaviour. These graphs track the evolution of the loss function over the course of training on both a 'train' and 'test' set, where the 'train' set refers to the set of collocation points used at that training step, and the 'test' set corresponds to a fixed, uniformly distributed set of points (not seen during training) used purely to evaluate the model accuracy. This provides insight on the convergence trend for each resampled set of collocation points and of the overall training, helping assess how adjustments to the resampling strategies influence the learning dynamics.

5.2.4 Memory Batching

Recalling the methodology described in earlier chapters, in this approach, a very large set of candidate points, $\mathcal{X}$, is generated randomly from a uniform distribution across the domain at each resampling stage. This is the set on which the selected guiding information is evaluated, and from which the probability density function

(PDF) is constructed, from which the new distribution of collocation points is selected.

While this approach was effective in the examined one-dimensional problems, this dense candidate pool resampling strategy incurs significant memory costs in higher dimensions. Since the candidate pool must be orders of magnitude larger than the number of collocation points N, the number of candidate points required for the dense grid to be representative grows exponentially with the dimension of the problem domain. This is because as N approaches $\mathcal{X}$ the closer the method is to being limited to the random resampling of the candidate set. In two dimensions, this approach is already impractical, as the higher end of the range of N investigated ($N = 100{,}000$) results in millions of candidate points being required. A test is carried out to assess the effect of the size of the candidate pool $\mathcal{X}$, shown in Figure 5.2. At the lowest $\mathcal{X}$, where $\mathcal{X} = N$ and we have effectively random resampling, the error is slightly higher than everywhere else across both depths. The decrease in error thereafter is not too significant, and is stable beyond $X \geq 12$ million points.

Twelve million points in this three-dimensional space-time domain represent a similar density to 100,000 points in the two-dimensional, 2×1 space-time domain we considered in Chapter 4 ($12{,}000{,}000^{2/3} \times 2$). The size of $\mathcal{X}$ selected in the two-dimensional problem was indeed 100,000, and so this 12 million point $\mathcal{X}$ represents over a tenfold increase in the size of the candidate set. Furthermore, at each point the guiding indicator evaluation now requires twice as many computations (one for each PDE) as in the single-equation case, increasing both computational cost and memory usage.

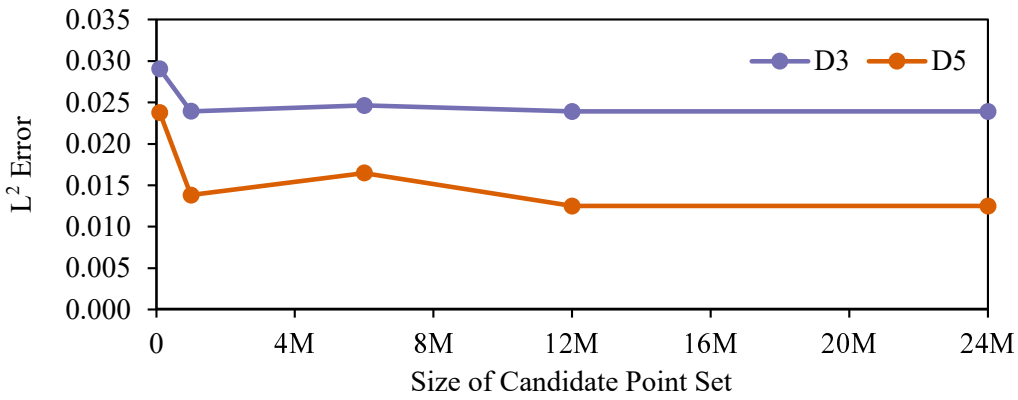

Figure 5.2: L_2 Error against size of candidate point set $\mathcal{X}$, using a residual resampling method, 100,000 collocation points, depths 3 and 5, and a 5-resample approach.

To carry out the aims set out for this chapter, a batching approach (Algorithm 3) was developed to circumvent the memory burden of evaluating the necessary gradi-

ents (associated with the specific guiding information that is being considered in each case) for this vastly increased number of candidate points at once, as this involves having to create large matrices which cannot be stored in memory with our given resources. The set $\mathcal{X}$ is split into smaller subsets, evaluating the chosen guiding information sequentially over each subset, with local PDFs computed and aggregated into a global distribution from which the final points were sampled. This approach minimises the number of large matrices created, whilst still saving the probability of every point in $\mathcal{X}$. This results in exactly the same probability being assigned for each individual point in the candidate set as if all were the candidate points had been loaded in memory at once.

Algorithm 3 Memory-Batched Candidate Pool Resampling

for $l = 1, \ldots, L_{\text{resamples}}$ **do**
    Generate new candidate pool $\mathcal{X}$ containing $\mathcal{X} >> N$ points
    Split $\mathcal{X}$ into subsets $\{\mathcal{X}^{(1)}, \ldots, \mathcal{X}^{(B)}\}$
    Initialise indicator vector $\mathbf{Y}$
    for $b = 1, \ldots, B$ **do**
        Evaluate the chosen guiding indicator on subset $\mathcal{X}^{(b)}$
        Store resulting values in $\mathbf{Y}^{(b)}$
        Append $\mathbf{Y}^{(b)}$ into $\mathbf{Y}$
    end for
    Construct the sampling PDF, $P(\mathcal{X})$, from $\mathbf{Y}$
    Sample N collocation points according to $P(\mathcal{X})$
    Update collocation set with sampled points
    Train until next resample
end for

5.3 Results

We first examine the cases where minimal changes were made to the model architecture and training regime from the 1D setup. Then the depth of the network is modified, as we expect the network's capacity to recreate the system of equations might be insufficient with the very simple $D = 3$ network, and observe the effect this has on the simple fixed method approach. Then we address adaptive resampling methods by varying the training approach, reducing the number of resamples whilst increasing the number of training steps performed on each set of points, keeping the total number of L-BFGS steps constant.

For all of the above, we show how the L_2 error of the model changes with number of collocation points N. For additional insight training loss histories are also shown for specific examples where this allows us to examine and discuss the underlying reasons for the performance observed.

Finally, using our observations from the previous investigations, we increase the value of ν to reduce the problem difficulty, and once again compare fixed and adaptive sampling methods. This comparison is again quantified through L_2 error versus N plots, cost tables, and specific selected loss history plots. A broad discussion of all the results is presented.

5.3.1 Original Approach

In the previous chapter, the model architecture used consisted of a fully connected network with a depth of 3 intermediate ('hidden') layers of a width of 64 nodes. For the system of two PDEs, the input and output layers have an additional input (x, y and t) and output (u and v). Other than these changes, the previous architecture is used for our initial tests in two dimensions, again solving the boundary conditions via an output transformation so that the loss function consists of only the residual loss calculated at the interior collocation points.

The optimisation strategy consists of a mix of Adam and L-BFGS steps: 15,000 initial Adam steps, followed by 50 resamples of 1,000 Adam and 1,000 L-BFGS steps. The reduction in number of resamples from the 1D tests was justified by the fact that loss and error had usually converged by that point, and the reduction in computational cost is highly significant for practical reasons due to the increased problem size for the higher dimension problem. The results obtained with this basic approach are shown in Figure 5.3. Adaptive methods all start with a Hammersley quasi-random distribution.

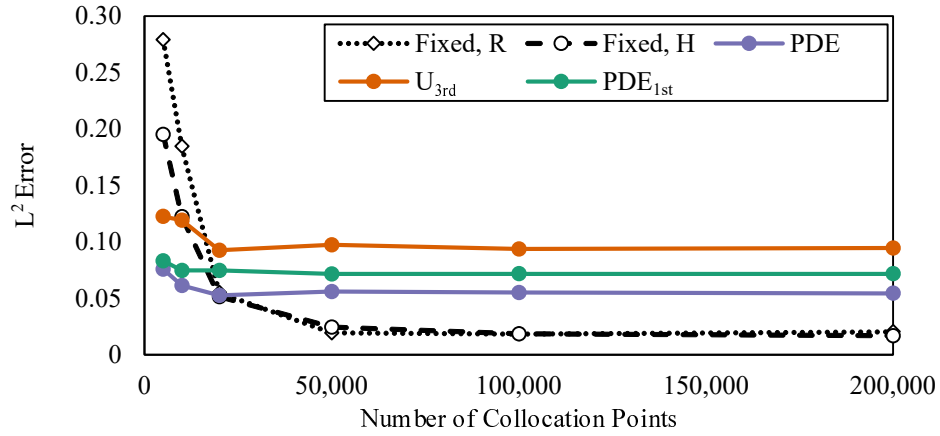

Figure 5.3: Error against number of collocation points for non-adaptive methods with random (Fixed, 'R') and quasirandom initial sampling (Fixed 'H'); and the main adaptive resampling methods outlined in section 5.2.3, using depth 3 and a 50-resample approach.

The error results shown in Figure 5.3 show fixed distribution methods converge

towards an error of approximately 2%. Meanwhile, adaptive resampling methods show a mostly flat error of around 5% - 10% as N increases. This high error using fixed methods, even as the number of points is greatly increased, indicates that the expressivity / capacity of our network may be too limiting. The adaptive methods' even worse performance also suggests that using the same training strategy (frequent resamples every 2,000 steps) does not continue to work in the new setup. By examining the loss history of two examples at $N = 20{,}000$ (before the minimum error has been achieved) and at $N = 100{,}000$ (when accuracy is no longer increasing with N) in Figure 5.4 we can get a better idea of how exactly this is happening.

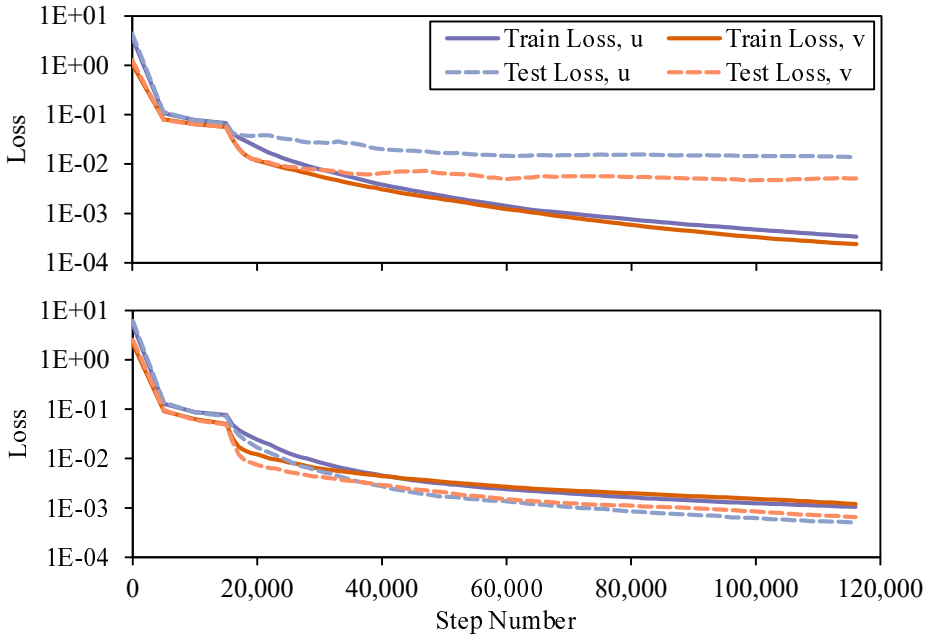

Figure 5.4: Loss history against steps, using a fixed Hammersley distribution and a depth of 3. Above: using 20,000 collocation points; below, $N = 100{,}000$.

In both cases, we see quick decrease in loss during the first 16,000 steps using the Adam optimiser, with behaviour differing after the change to L-BFGS. For the case with lower N (above), we observe the loss on the training set of points continue to decrease throughout training, whilst the loss at the uniform (unseen) test set quickly stops improving. Seemingly, the training set does not cover the entire domain well enough that the network's performance at the training test generalises. The opposite is true in the higher N case (below), where driving down the loss at the training set of points results in a NN that can easily drive down the loss evaluated over the uniform test set. In both cases, the lowest losses achieved are in the range of 10^{-4}, which are still too high to signify convergence to errors below 10^{-3} that we observed in the 1D case.

To address the possibility that this is an issue with network capacity, we next

show the results of increasing the network depth for these fixed methods. In Figure 5.5, the error against number of collocation points is shown, for network depths of $D = 5$ and $D = 8$, using fixed distributions of points. We see quasi-random distributions again outperforming random initialisations across all depths. At very high N, the deeper networks achieve errors under 10^{-2}, with $D = 5$ achieving 0.9% at $N = 100{,}000$ and $D = 8$ achieving 0.8% error at $N = 200{,}000$. However, using a depth of 8 substantially increases the number of parameters to train, presumably making the network harder to train and causing a worse performance at the intermediate point ranges $20{,}000 \leq N \leq 50{,}000$ than the shallower depth networks.

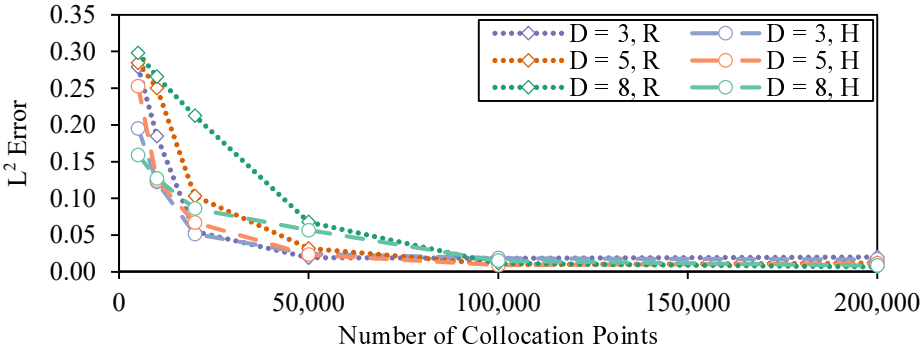

Figure 5.5: Error against number of collocation points using fixed distribution methods only with random ('R') and Hammersley ('H') initial point distributions, across network depths 3, 5 and 8.

Next, we examine two loss history graphs obtained using the original residual resampling method, with 50 resamples and a shallow depth of 3, for two numbers of points, $N = 20{,}000$ and $N = 100{,}000$. This is shown in Figure 5.6. We observe that unlike in 1D, the increase in loss due to resampling occur so frequently, that the loss does not decrease enough within each resample to offset the increase at the start of subsequent resamples. As such, the training losses don't trend downwards sufficiently and remain high, indicating that progress towards finding a good solution isn't occurring. In the fixed methods example shown in Figure 5.4, at low N we observe a kind of overfitting occur where the test losses were not driven down by training. That behaviour is not observed in this scenario, but the training is more erratic, with multiple increases to test loss observed in some of the resamples. Around 95,000 steps, a series of resamples cause the v test loss to greatly increase, as the optimiser seems to 'unlearn' previous information, and it takes a few resamples for the loss to decrease back to the previous level. In contrast, in the case with a greater number of collocation points we can observe the losses to more smoothly trend downward. However, this training loss still plateaus at a high value around

0.01 (compared to 1×10^{-6} losses observed in the 1D case), with the test loss and error not actually decreasing significantly with N.

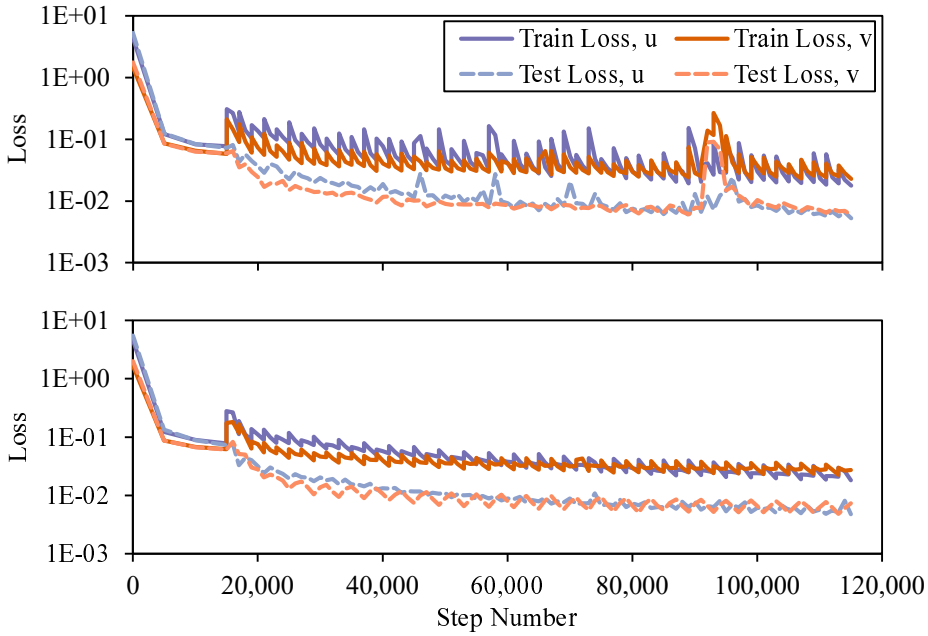

Figure 5.6: Loss history against steps, using residual resampling and a depth of 3, with 50 resamples. Above: using 20,000 collocation points; below, $N = 100{,}000$.

To visualise what these error magnitudes look like, Figure 5.7 shows some spatial slices of u at certain time points T. At a glance the differences between the ground truth (top row) and PINN prediction (middle row) are not obvious, with the shapes and locations of the shock being fairly accurate. From the contour (bottom row) we can see that the regions with highest error are the top corners at lower T, and around $x = 0$ at higher T, where at a close glance the PINN prediction of the magnitudes of u can be observed to be slightly wrong, predicting a less sharp 'shock'. Figure 5.8 shows the same slices of v, and this time the error is seen to concentrate around the top boundaries and corners at early T. Outside these high error regions, error is consistently around 10^{-2} throughout the domain. A visualisation of the overall point distribution (Figure A.2) shows that whilst adaptive resampling identifies the middle shock region and assigns some collocation points to that region, comparatively more points are assigned to adequately resolving the boundary regions at the top corners.

Based on all the observations shown in this section, we can see that broader adjustments need to be made for the approach and network to enable it to work for the higher dimensional system of equations. The adjustments to the network parameters don't allow the baseline fixed distribution method to achieve accurate results by themselves, and so there is still a gap to be filled by using adaptive resampling. From our post-processing of the loss history, the logical next step is

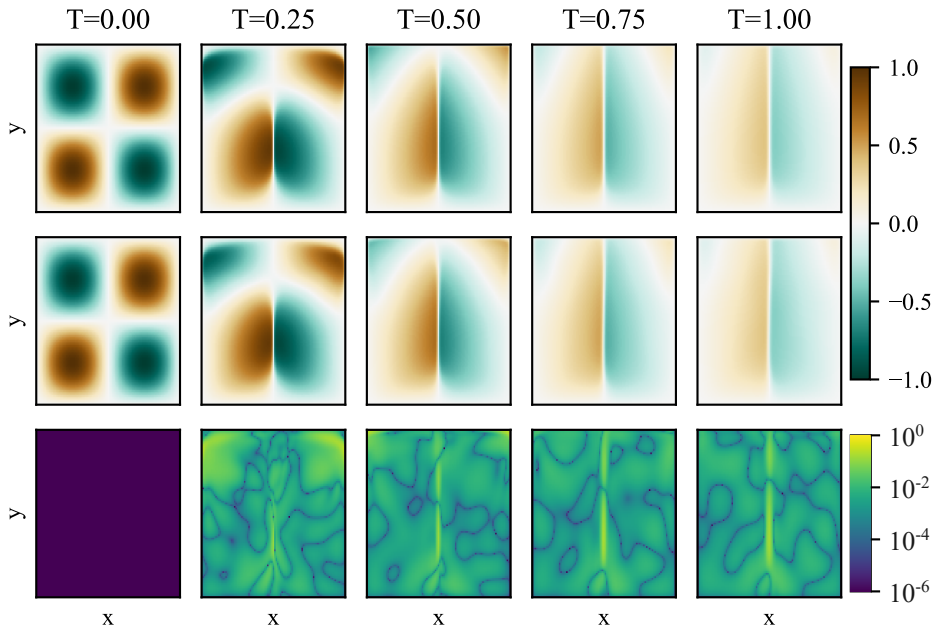

Figure 5.7: Above, ground truth value of u; middle, PINN prediction; below, absolute error. Obtained using a PDE method with 20,000 collocation points, depth 3 and 50 resamples. The L_2 error for u in this run was 4.80×10^{-2}.

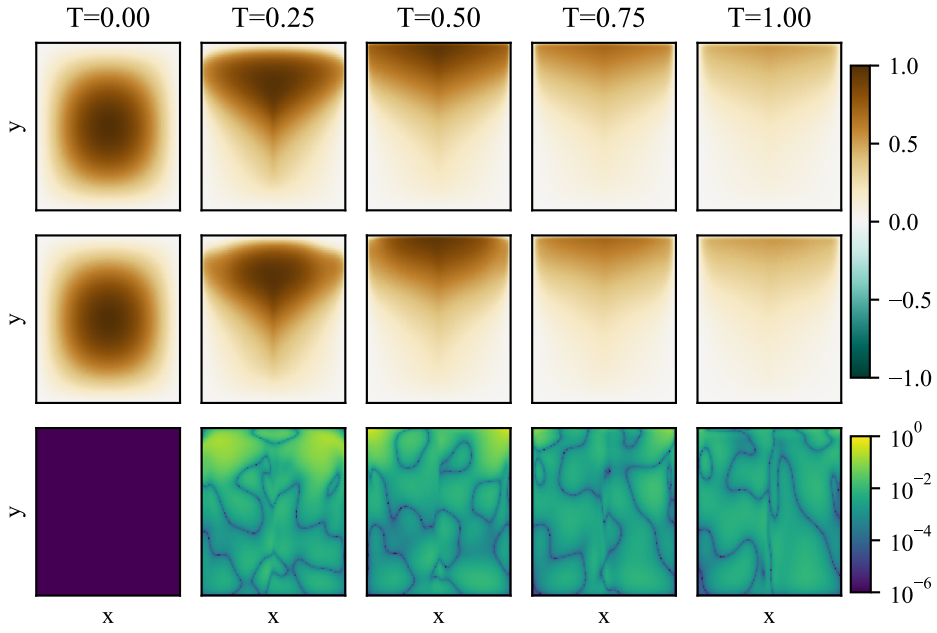

Figure 5.8: Above, ground truth value of v; middle, PINN prediction; below, absolute error. Obtained using a PDE method with 20,000 collocation points, depth 3 and 50 resamples. The L_2 error for v in this particular run was 6.45×10^{-2}.

to adjust the adaptive training strategy to use longer, but less frequent, resamples. The process of selecting that adjusted approach is the topic of the following section, where a range of resample frequencies is tested.

5.3.2 Resample frequency

In the previous section it was observed that the current resampling strategy, characterised by a large number of resamples with 2,000 steps optimisation steps, may be limiting the effectiveness of adaptive resampling in the unsteady two dimensional case, with the set of collocation points being replaced before the optimiser has been able to effectively learn from the current set. In this section, we present tests to determine the appropriate number of optimisation steps per collocation point distribution ('per resample'). We then set the number of resamples accordingly, with the objective of keeping the total number of training steps comparable between cases. The decision is made to keep the total number of L-BFGS steps fixed, rather than the total number of all optimisation steps (see table 5.1), for two main reasons. Firstly, the L-BFGS steps are more computationally expensive and therefore are a more significant contribution to the computational cost. Secondly, the role of Adam steps is to act as a 'rough' initial optimiser to help initialise training, and so the number of Adam steps doesn't contribute to the final, lowest error obtained by the network.

To initiate this study we consider 4 configurations. Table 5.1 summarises these configurations in terms of the number of resamples L, detailing the number of Adam and L-BFGS steps per resample, as well as the total L-BFGS steps and overall optimisation steps. This information provides the necessary context for interpreting the results of the subsequent analysis. These configurations are all tested using residual based adaptive resampling method with two neural network depths, shown in Figure 5.9. There we plot the final L_2 error against the number of collocation points.

L	Adam Steps	L-BFGS Steps	Total L-BFGS Steps	Total Steps
5	1000	20,000	100,000	105,000
10	1000	10,000	100,000	110,000
25	1000	4000	100,000	125,000
50	1000	2000	100,000	150,000

Table 5.1: Table showing the different configurations of number of resamples, L, and corresponding number of optimisation steps per resample, and total steps. The number of L-BFGS steps per resample is adjusted inversely with L to maintain a constant number of total L-BFGS steps.

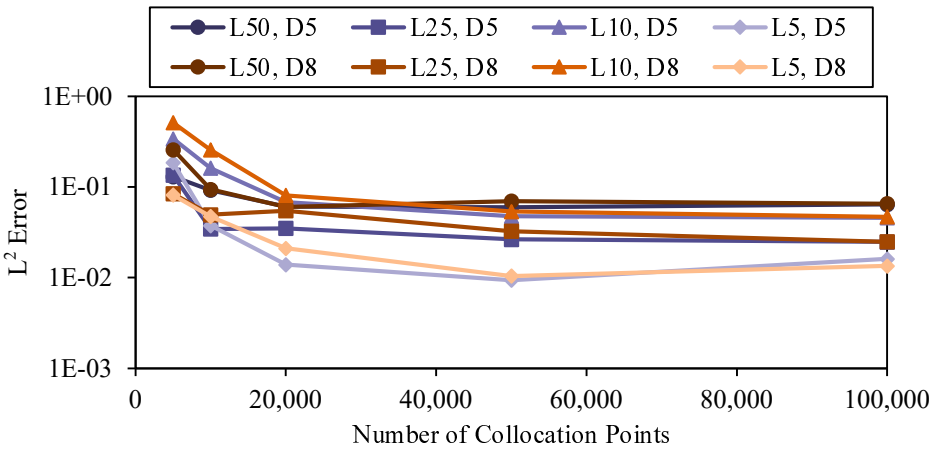

Figure 5.9: L_2 Error against number collocation points, using residual resampling, and depths of 5 and 8. The different curves represent different configurations of number of resamples and optimisation steps as outlined in table 5.1.

Inspection of the curves in Figure 5.9 suggests a number of general trends. Across all cases, error decreases initially as the number of collocation points is increased from 5,000 to 20,000, generally reaching close to the highest accuracy around $N =$ 20,000 or 50,000. This lowest error (9.38×10^{-3}) is achieved by using the $L = 5$ approach, using 20,000 L-BFGS steps in each resample with a depth of 5. This represents a 5 fold decrease in error compared to the previous lowest error (5.25×10^{-2}) achieved with a depth of 3 and the 50 resample approach considered previously (See Figure 5.3). Furthermore, using a depth of 5 mostly achieves similar or lower error (to the depth 8 cases) across all four of the configurations tested. An increase in error in the $L = 5$ case at $N =$ 100,000 is also observed, and we speculate that despite the increased domain coverage, the large amount of collocation points may actually be increasing the complexity of the optimisation problem.

Based on these results, the last configuration with only 5 resamples is deemed the most appropriate and so in following sections this will be used as the default implementation of adaptive resampling. With this decided, the next step then is to expand the error and cost investigations to more of the guiding information methods we introduced in the previous chapter for the 1D Burgers' Equation.

5.3.3 Modified Approach

So far in this results section we have tested modifications to our previous approach from chapter 4, and these suggest using deeper networks and a modified training approach when applying adaptive resampling methods (based upon tests using the local residual). Using these adaptations, we now present further investigations, showing the error and also the computational cost associated with different resam-

pling indicators.

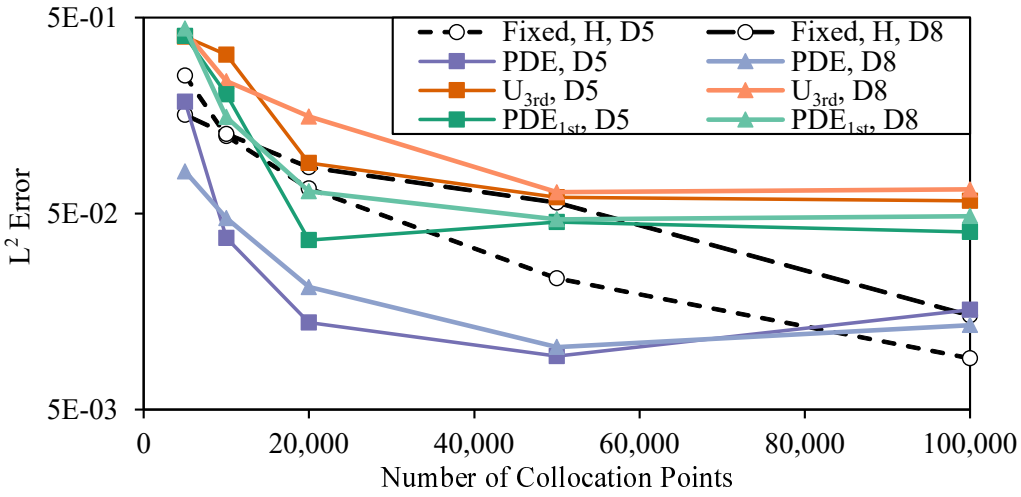

Figure 5.10: L_2 Error against number collocation points using different adaptive resampling methods, varying depths, and a 5-resample approach.

First, the error against number of collocation points is shown in Figure 5.10, where the fixed method baseline is compared to adaptive resampling with different information sources guiding the resamples. Broadly, we observe the best errors are still obtained by an adaptive approach, namely residual-based resampling at $N = 50{,}000$. The non-adaptive approaches do better with increasing N, but not reaching comparable errors until $N = 100{,}000$. Overall, errors are lower than in the original approach in Figure 5.3, but derivative metrics perform poorly in comparison to the other methods. We again observe that adaptive resampling methods don't always show a convergence towards a minimum error as N is increased, but instead the error re-increases with N past an optimum point.

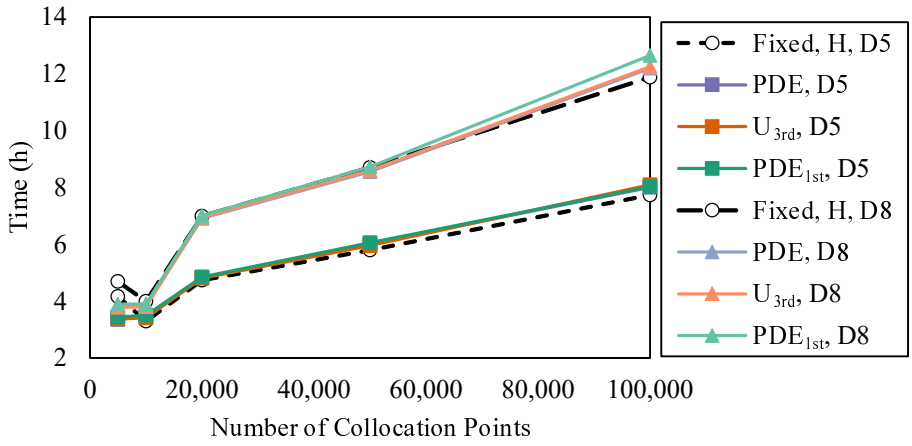

Figure 5.11: Computational cost in hours against number collocation points using different adaptive resampling methods, varying depths and a 5-resample approach.

Next, the computational cost of the new approach is shown in Figure 5.11.

The main observation is that the choice of method has a relatively minor impact compared to the number of collocation points and neural network depth. Past $N = 10{,}000$, the runtime scales approximately linearly with N, with a slightly steeper gradient for the depth 8 case. The two sets of curves corresponding to different depths are separated by a significant offset, where the increased depth is more costly for all choices of N, most significantly from $N = 20{,}000$ onwards. A final observation is that for the smallest number of collocation points tested, $N = 5{,}000$, the fixed distribution method requires more time than at $N = 10{,}000$, suggesting that an excessively low number of collocation points can increase training time due to poor convergence.

To compare with the loss histories previously shown in figures 5.6, we next examine a loss history obtained from the most successful adaptive resampling method from Figure 5.10. This is shown in Figure 5.12. From the loss history we can see how the longer training stage for the first resample (at 16,000 steps) already allows the losses to drop lower than has been observed in the previous examples shown in this chapter (below 10^{-3} by 36,000 steps). Subsequent resamples cause sharp increases in the loss as the network readjusts to the newly selected collocation points. However, the subsequent optimisation steps do not reliably reduce the loss below the minimum achieved by the previous resample. As a result, a steady downward trend in the loss is not observed unlike in the unsteady 1D case, where each resample caused a brief increase in the loss followed by further reduction to a lower minimum before the next resample. Instead, improvements in accuracy depend on individual resamples in which the points enable the optimiser to drive the loss down to a low minimum.

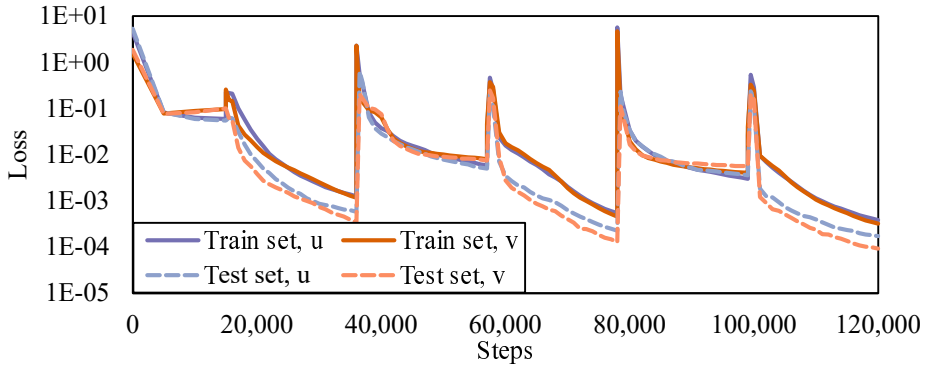

Figure 5.12: Loss history obtained using residual resampling, 50,000 collocation points, and a depth of 5.

For another form of post-processing,the concentration of error is shown again in figures 5.13 and 5.14, to illustrate which areas of the solution the PINN struggles

to approximate accurately. For u, the high-error regions previously observed in the corners are now largely resolved, appearing only around the $x = 0$ line at later values of T. Outside these localised regions, the error is noticeably lower, with values falling below 10^{-3}. Regarding v, only small areas near the boundary corners around $T = 0.5$ exhibit errors above 10^{-2}.

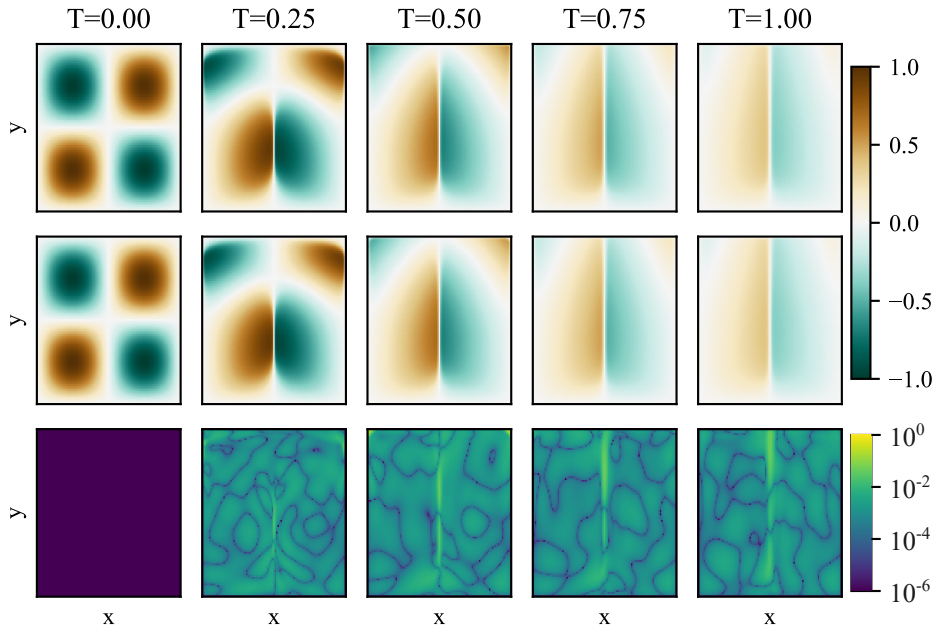

Figure 5.13: Above, ground truth value of u; middle, PINN prediction; below, absolute error. Obtained using a residual resampling method with 50,000 collocation points, depth 5 and 5-resample approach. The L_2 error for u in this run was 1.08×10^{-2}.

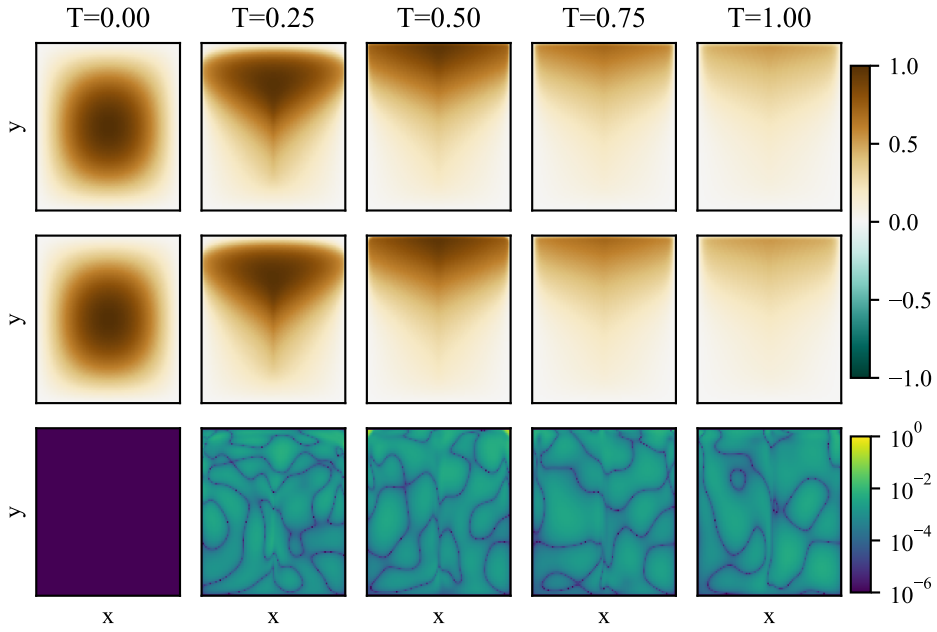

Figure 5.14: Above, ground truth value of v; middle, PINN prediction; below, absolute error. Obtained using a residual resampling method with 50,000 collocation points, depth 5 and 5-resample approach. The L_2 error for u in this run was 6.52×10^{-3}.

5.3.4 Higher ν case

Solving the same PDE system (Equations 5.1, 5.2) but with a higher ν results in a visually similar solution, with a wider 'shock' in the u component and a wider boundary layer in the v component (both of which dissipate more rapidly in time) due to the increased viscosity. This should result in the solution being easier to compute/learn. Using the approach with less resamples ($L = 5$), we present in Figure 5.15 the error obtained from multiple tests, with $\nu = 0.01$, against the number of collocation points, N.

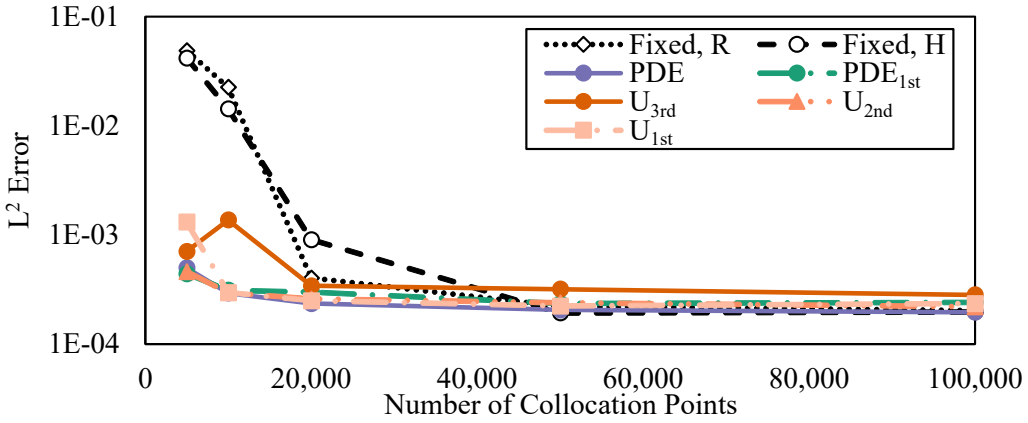

Figure 5.15: For $\nu = 0.01$ - Error against number of collocation points obtained using a NN depth of 5 and 5-resample approach.

This case shows accuracy levels similar to those previously observed in the 1D case in Chapter 4, with all methods converging to an order of 10^{-4} with sufficient points, despite the increased problem dimensionality and additional PDE. Fixed distribution point methods show high errors below $N < 20{,}000$, whereas all of the adaptive methods considered lead to errors around or under 10^{-3} for the lowest value of N tested (5,000). Above $N = 20{,}000$, the error obtained by the different adaptive criteria is indistinguishable, aside from the third-derivative of solution indicator U_{3rd}, which remains higher by approximately 1×10^{-3}. At $N = 5{,}000$, U_{2nd}, PDE and PDE_{1st} show very low errors of 4.5×10^{-4}, only a fourth the amount of N required by the non-adaptive quasi random case to reach the same error.

A qualitative assessment of these reduced errors is provided in figures 5.16 and 5.17, which present the error contours using the residual resampling method, as in the previous section, but using fewer collocation points. The solution of u shows a broader velocity transition in the x-direction, which could explain the absence of a high-error region along the $x = 0$ line observed in earlier cases. Overall, the error across the whole domain is notably lower, consistently ranging between 10^{-3} to 10^{-6}. The solution in v also shows a broader boundary layer at the top boundary, with the corresponding error contour showing low error along the top boundary and throughout the domain.

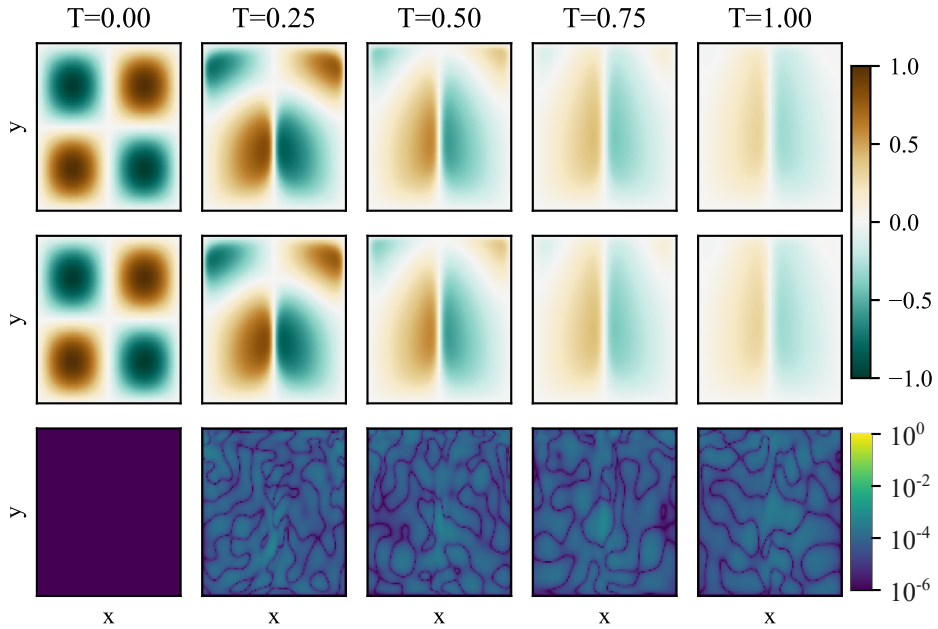

Figure 5.16: For $\nu = 0.01$ - Above, ground truth value of u; middle, PINN prediction; below, absolute error. Obtained using a PDE method with 20,000 collocation points, depth 5 and 5-resample approach. The L_2 error for u in this run was 2.52×10^{-4}.

The final assessment is of the training loss, presented in Figure 5.18, for compar-

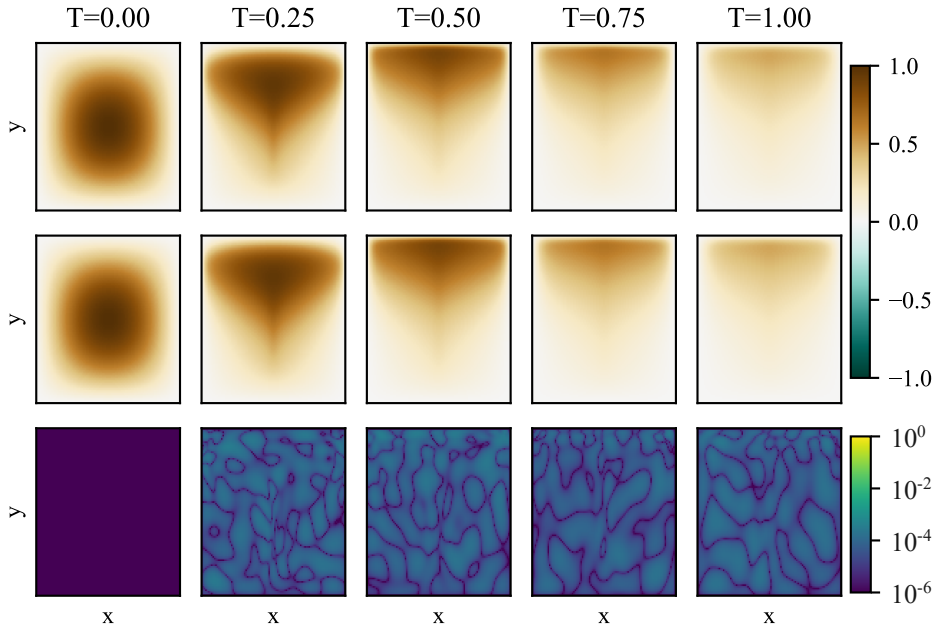

Figure 5.17: For $\nu = 0.01$ - Above, ground truth value of v; middle, PINN prediction; below, absolute error. Obtained using a PDE method with 20,000 collocation points, depth 5 and 5-resample approach. The L_2 error for u in this run was 1.79×10^{-4}.

ison with Figure 5.12. The benefit of using fewer, longer resamples is clearly evident here, with the loss approaching a plateau before each resample. As in previous cases, spikes in training and test error occur as the collocation points are updated and the network adjusts. However, the losses progressively decrease across resamples (from 9.7×10^{-6} after the first to 1.9×10^{-6} at the last) indicating steady overall improvement.

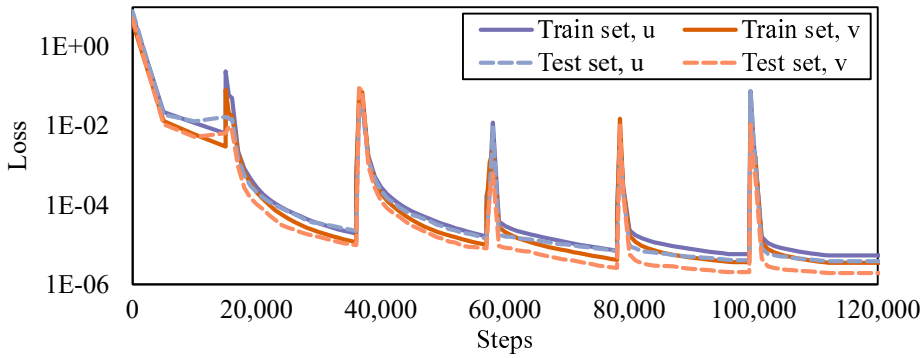

Figure 5.18: For $\nu = 0.01$ - Loss history example, obtained using 10,000 collocation points, depth 5 and 5-resample approach.

We now consider the fixed case, which as seen in Figure 5.15, produces a substantially higher error until the collocation points are sufficiently increased to $N =$ 20,000. To investigate this behaviour, Figure 5.19 presents the corresponding loss histories for $N = 10{,}000$ and 20,000, obtained using a fixed quasi-random sampling method and a neural network depth of 5. In both of these loss graphs, we see the ini-

tial training phase with Adam up until 16,000 steps, when the optimiser is changed to L-BFGS, at which point the latter causes a sharp change in the loss gradient.

In the $N = 10{,}000$ case, while the training losses for both u and v decrease to between 10^{-5} and 10^{-6} after around 50,000 steps, the test losses after the initial training with Adam jump to and remain at around 10^{-2}, showing a clear discrepancy between the network's performance on the training points and on the test set. This looks like a typical example of overfitting to the collocation points when there are insufficient of them. In contrast, for $N = 20{,}000$, all loss curves drop to between 10^{-5} and 10^{-6}, with the test losses remaining slightly below the training ones throughout.

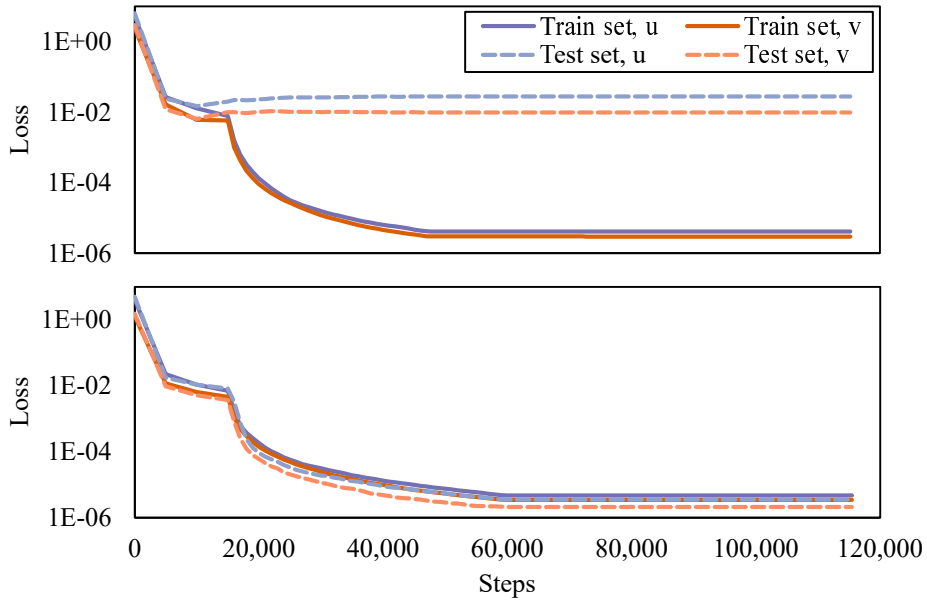

Figure 5.19: For $\nu = 0.01$ - Loss history against steps, using fixed methods, $N = 10{,}000$ (above) and $N = 20{,}000$ (below), with a NN depth of 5 layers.

Finally, we examine the time costs in Figure 5.20, fixing the network depth to 5 layers. The difference between methods for a given N and D in time taken are very minor, compared to the changes across those parameters. The fixed distribution cases took slightly longer, which is not totally unexpected behaviour, as the larger proportion of expensive $L - BFGS$ optimisation steps can easily contribute more to the computational cost than the costs associated with computing guiding information for resampling. All cases scale approximately linearly with number of points, which is the main factor in the time taken.

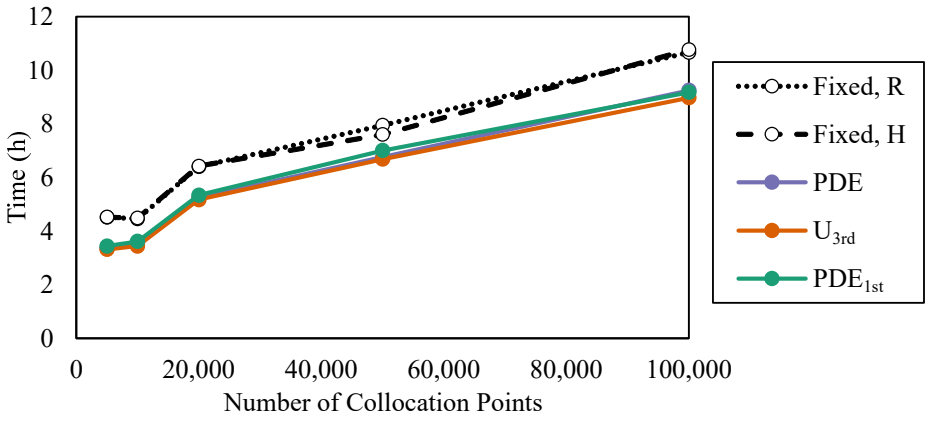

Figure 5.20: For $\nu = 0.01$ - Computational cost of training against number of collocation points, for a NN depth of 5 layers.

5.4 Discussion

5.4.1 Overview and General Remarks

This chapter extends our investigation of adaptive resampling techniques to a two-equation system solving the two-dimensional Burgers' equation. The aim of this chapter is to evaluate the potential benefits previously identified that help the scalability of PINNs, by applying those methods a more complex problem (a PDE system in a higher dimension). To maintain consistency the PINN has been adjusted to solve a two system of equations, but the resampling strategy has been kept the same.

Adapting to the higher-dimensional setting required changes to the guiding indicators. In addressing the dimensional aspect of the problem, an important question arises on the scalability of using increasingly higher-order derivatives as guiding indicators, as we did in the previous chapter. In particular, computing the third order derivative of the residual with respect to all dimensions was not possible due to memory requirements of the operation. As a result, the equivalent of the PDE_{xt} method in 1D (which showed significant benefits to accuracy and N reduction), PDE_{xyt}, is unfeasible for the current problem, and so the investigation had to focus on whether similarly positive results to those of the PDE_{xt} method in Chapter 4 can be obtained limited to combinations of residual first order derivatives ($\partial_x PDE$, $\partial_y PDE$, $\partial_t PDE$).

The increase in dimensionality also expands the size of the space-time domain, and the additional complexity from now dealing with a system of equations demands for better coverage. A consequence of this is that the size of the candidate set of points, $\mathcal{X}$, must be increased proportionately to the increase in the number of collocation points N needed to cover the new domain. In particular, to maintain the

same density as in the unsteady 1D setting requires scaling $\mathcal{X}$ from N to $N^{3/2}$. This is needed so that the point selection does not reduce to random-resampling, as when N approaches $\mathcal{X}$, all the points in $\mathcal{X}$, which are distributed randomly, are selected regardless of the PDF obtained. However, increasing the size of $\mathcal{X}$ introduces two challenges. First, evaluating the guiding indicators and residuals at more candidate points incurs higher computational cost. Second, as noted in Hanna et al. [2022], as $\mathcal{X}$ increases the risk grows of oversampling regions with high residuals or sampling only the highest residual regions, leading to less effective coverage. And whilst the latter challenge can be addressed in our current approach by the tuning of the PDF hyperparameters to increase base probability (via c) and reduce scaling (k), the former presents a challenge to scalability in regard to both execution time and memory requirements.

The initial adaptation of the adaptive resampling method without changes to training strategy (fig. 5.3) showed high errors relative to what we have previously observed, which did not decrease as more collocation points were used. Through the initial investigations, a minimal increase to network depth and a more substantial alteration to training strategy were used that did achieve more satisfactory results at a qualitative level, achieving accuracies under 1×10^{-2} (fig. 5.10). Whilst in comparison with results from the previous chapter the errors from this 'modified' approach are around an order of magnitude higher, the comparison with the fixed baseline showed that once again certain adaptive resampling indicators (the pure residual in this case) can be used to achieve similar accuracies ($< 1\%$) to fixed methods whilst using significantly fewer points (20,000 vs 100,000). Within the adaptive methods however, only the residual-based criteria seemed reliable, with the residual first-derivatives and solution-curvatures showing minimum errors around 5%.

When a simpler variant of the system of PDEs was considered using $\nu = 0.01$ in Section 5.3.4, however, results more consistent with those of Chapter 4 were observed. Here, all adaptive resampling methods (except for the third derivatives of solution, U_{3rd}) were able to achieve errors under 1×10^{-3} using only $N = 10{,}000$, whist non-adaptive methods required up to 20,000 and 50,000 points depending on the initial sampling strategy. An interesting observation was that the minimum error achieved by the U_{3rd} case was slightly higher than other indicators throughout the range of N, which was also the case for the lower ν investigations. This perhaps indicates an suboptimal selection of PDF hyper-parameters, or simply that this indicator may not be well-suited to this particular problem.

In chapter 4, adaptive resampling methods outperformed the non-adaptive counterparts, with the high-order residual derivative indicator PDE_{xt} frequently achieving the lowest errors. Although an equivalent to this high-order residual derivative indicator is not available for the problems considered in this chapter, we still observed other indicators can drive adaptive resampling methods to perform better than non-adaptive approaches when the training strategy is appropriately adjusted (Figure 5.10). Overall, this suggests that the advantages of adaptivity do not disappear even as the number and dimensionality of the PDEs increase, Overall, this suggests that while adaptivity cannot reduce the higher error of PINN when solving the more challenging high viscosity case, it can still present a small improvement over non-adaptive resampling.

5.4.2 Limitations and Next Steps

The resampling method alone is not sufficient to enable PINNs to scale effectively, as other aspects of the training process remain insufficiently robust under the current approach. In particular, the fact that many resamples fail to improve on the minimum loss achieved by previous resamples highlights that the finer details of how resampling is performed become increasingly important as problem complexity grows. At present, fixed methods offer the advantage of simpler implementation, avoiding the need for additional hyperparameters or the design of a resampling strategy. However, they do not achieve sub-1% accuracy (in the $\nu = 0.01/\pi$ case) or sub-0.1% accuracy (in the $\nu = 0.01$ case) without using significantly more N than adaptive resampling methods. As such reductions in the number of collocation points required may ultimately be necessary to ensure scalability of PINNs, more research into improving adaptive resampling is necessary.

Some improvements like implementations of annealing schedules have already been seen in the literature Subramanian et al. [2023] and may be adapted to help automate some of the aforementioned issues with automating model design and resampling. Other broader enhancements to automating PINNs design are also likely to emerge in the near future, and so we move on to discussing the issues we observed with training stability specifically in the case of our adaptive resampling.

Building on the observations made in this chapter relating to training loss behaviour, we can identify several areas for further investigation. For instance, Figure 5.12 illustrates an important issue of training loss not consistently decreasing across resamples, and, in some cases worsening. This implies that the network is not able to preserve the progress made in earlier training stages when transitioning

to a new set of collocation points.

Several causes may underlie this behaviour. One possibility is that the network architecture is not sufficiently robust to a sudden, complete change in the collocation point set, which implies that full replacement of collocation points as a method may be too disruptive in challenging problems. Some alternative approaches in the literature retain a subset of points between resamples (such as Zapf et al. [2022] and Daw et al. [2023], which add and remove only a fraction of the total points at a time), which could help maintain continuity in training for our method. Extending this, trying to keep not just a random subset of points, but particularly informative training points, could help improve stability and preserve focus on the more complex features.

It is important at this point to emphasise that collocation points do not act as a conventional training set, as they define the loss function itself. Hence, resampling makes a discontinuous change in the objective of the minimisation midway through the optimisation process. This means the network may be struggling not just with data continuity, but with a change in the loss landscape. We could thus also consider evaluating the optimisation process itself. The sudden change in collocation points may cause the optimisers to effectively reset the network's weights, rather than help fine-tune them. This could reflect a need for an optimisation scheme more robust to these dynamics, possibly by incorporating adaptive learning rates or momentum mechanisms to better handle the abrupt shifts.

Whilst these last observations point towards specific limitations of the current approach, we have also established the potential of our methods to reduce computational cost by achieving higher accuracy with fewer collocation points. These results provide a strong motivation for the next chapter, which investigates some of these issues in greater depth and tests strategies aimed at improving the robustness of PINN training in our two-dimensional problem.

Chapter 6

Improvements to Higher-Dimensional case

In this chapter, we introduce three modifications to the resampling methods we employed in the previous chapter. These are motivated by approaches found in broader literature. Our aims with this are first, to assess whether we can improve the performance of our introduced framework in the 2D setting, and second, to gain more information on what areas of our method require refinement for further improvement with regards to the scalability of accuracy and cost of our methods.

Section 6.1 presents the objectives of this chapter and explains the rationale for the proposed modifications. The subsequent methods section provides a detailed description of the three strategies introduced. In the results section, we report the investigations conducted for each approach. Finally, Section 6.4 offers a discussion of the findings.

6.1 Introduction

In the work presented in the previous two chapters, it is demonstrated that adaptive resampling strategies can significantly aid PINNs in solving unsupervised learning problems, enabling models to achieve lower errors while often requiring fewer computational resources in the form of collocation points (see Chapter 4). As the dimensionality of the problem increases, so does the volume domain, creating a need for strategies that help reduce the scaling cost of PINN training, such as our adaptive strategies. It is also observed that the effectiveness of adaptive resampling is strongly influenced by how adaptive resampling is implemented (see Chapter 5).

In the previous chapter, the implementation of resampling is focused on hyperparameter tuning to maintain accuracy and cost-effectiveness as dimensionality

increases, specifically by adjusting the frequency of resampling and the network depth. In this chapter, attention is shifted to adaptations that directly influence the treatment of collocation points. To this end, three approaches are examined that are designed to address underlying factors with a potential to affect model training and performance.

The first factor to be investigated is whether high variance in the sampling of collocation points between resamples hinders training, as previously only the entire set of collocation points has been resampled at once. The first adaptation is implemented by separating the collocation points into two sets, only one of which is adaptively resampled, while the second remains as a fixed quasi-randomly distributed set (a technique that is motivated by similar approaches reported in Hou et al. [2023], Chen et al. [2025]). This separation allows adaptive resampling to focus on regions of high residuals or gradients, while the second set ensures adequate overall coverage.

A second consideration is the resampling of collocation points in systems of multiple PDEs. When residual or gradient information from all equations is aggregated, it is hypothesized that resampling based on this combined information may relocate points to regions that are suboptimal for individual equations. Furthermore, variations in the magnitude of residuals and gradients between PDEs could cause some equations to exert less influence on the resampling process, regardless of their underlying importance. To address this, a targeted strategy is investigated in which collocation points are divided into subsets that are resampled independently with respect to the residuals and gradients of each PDE. The closest related work is Hanna et al. [2022], where a PINN is partitioned into separate subnetworks for each target output. While the present approach differs in that the sampling rather than the network architecture is partitioned, both methods aim towards direct learning per equation. This design enables an assessment of whether equation-specific resampling can enhance convergence and allow more effective point management.

The third method combines the two strategies described above. The collocation points are partitioned into a fixed quasi-random subset that is never resampled, providing baseline coverage of the domain, and additional subsets per equation that are resampled independently according to the residuals or gradients of their associated PDE. This design aims to direct sampling pressure toward equation-specific deficiencies while simultaneously controlling variance through the fixed subset. In principle, the relative sizes of these subsets could be adjusted over the course of training to balance global coverage and per-equation focus; however, the design and validation of such adaptive allocation are beyond the scope of this chapter.

In addition, a separate strategy inspired by dropout regularisation (Srivastava et al. [2014]) is tested, in which at each resampling stage a random subset of points is excluded from resampling and retained in place, with the aim of further smoothing transitions in the training process.

6.2 Methods

A coupled system of PDEs describing Burgers' Equations in two dimensions evolving in time is solved, as described in the previous chapter (see Section 5.2). The same initial and boundary conditions are used, with a viscosity $\nu = \frac{1}{100\pi} \sim 0.003$.

The same underlying PINN fully-connected network and optimiser selections described in Section 5.2 are used, implementing the batching technique from Section 5.2.4 and the resampling strategies investigated in 5.3.3, using a network depth of 5 and resampling collocation points every 20,000 optimiser steps.

6.2.1 Fixed and Adaptive set partitioning

The first strategy separates the collocation points into two subsets: one that follows adaptive resampling and another that remains fixed throughout training. These broad ideas of applying different strategies to subsets of points have been explored in different ways, such as recent work in Chen et al. [2025] and relatively older work in Hou et al. [2023]. The novelty of the strategy introduced in this subsection is the use of a quasi-random distribution, rather than uniform random sampling, for constructing the fixed subset. Quasi-random point sets have been shown to provide improved domain coverage relative to purely random samples (see Chapter 3 and Wu et al. [2023]), which may allow comparable accuracy to be achieved with fewer points. Ideally, this is also more effective than relying on the random placement in adaptive resampling, guided by the hyperparameter c (Equation 2.1). The downside of a fixed grid however is the model gets to train on less potential locations. To assess the extent of this effect, a variant procedure is implemented and tested, in which the fixed quasi-random grid is randomly translated at each resampling step (similar to Chen et al. [2025]). This maintains the domain coverage characteristic of quasi-random distribution whilst allowing the model to train in more individual locations.

As for the implementation, the full set of N collocation points are distributed quasi-randomly for the initial 15,000 Adam steps. At the first resampling stage, a specified proportion of the points is allocated to the fixed subset, and the remain-

ing form the adaptive subset. The first subset is sampled following a quasi-random algorithm to ensure that coverage is conserved, and is unchanged through subsequent resamples. Meanwhile, the adaptive subset is resampled normally according to the guiding information obtained, replacing points from the previous adaptive subset. For individual optimisation steps in between resamples, information from both subsets is treated equally for training.

In the hybrid approach, the difference is implemented at the subsequent resamples. Instead of the fixed-subset points remaining unchanged, all points undergo an equal translation, a random shift in each spatial direction within the domain interval $[0, 1]$. Any translated point that lies outside the domain is wrapped back by subtracting the domain length, thereby ensuring that all points remain within the problem domain.

6.2.2 PDE based partitioning

The second resampling strategy divides the point-resampling operation according to the information from each PDE equation individually. In this approach, a subset of points is resampled using information derived from the equation for the evolution of the first solution component (u) and its associated residuals or derivatives, whereas the second subset of points uses the equation for the evolution of the second solution component (v) and its corresponding residual or derivatives.

To implement this approach, first a fraction is manually specified at the beginning of training to decide the size of each subset. The training procedure remains unchanged, with modifications applied to the resampling stage. At the start of resampling, two probability distribution functions (see Eq. 4.7) are computed: the first using $Y = Y_u$ derived from the PDE governing u, and the second using $Y = Y_v$ derived from the PDE governing v. As an example, using residual information for resampling Y_u and Y_v may be expressed as:

$$Y_u = \frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} + v\,\frac{\partial u}{\partial y} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{6.1}$$

$$Y_v = \frac{\partial v}{\partial t} + u\,\frac{\partial v}{\partial x} + v\,\frac{\partial v}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \tag{6.2}$$

A specified fraction of the points is selected according to the first probability distribution function (PDF), and the remaining points are selected according to the second PDF. The combined set is then treated as a single collection for the continuation of training, with all points assigned equal weighting.

When the points are separated according to the two PDEs, the interaction with the candidate-grid selection procedure introduces a minor effect worth noting. Under the standard approach, the residual is evaluated across the entire candidate grid to form a single PDF. In the partitioning strategy, two PDFs are generated by applying the same procedure twice on the same dense grid: once for Y_u and once for Y_v. As a consequence, individual points may be selected in both subsets. This remains unlikely due to the grid density and differences in the resulting probability distributions. However, even if duplication occurs, it does not adversely affect the method. The corresponding residual information is simply used twice during weighting, effectively assigning greater weight to locations that are influential for both PDEs, a behaviour that may be interpreted as beneficial rather than problematic.

6.2.3 Fixed and PDE-based set partitioning

A third strategy combines the fixed–adaptive partitioning with the PDE-based partitioning described in the previous subsections. In this approach, the collocation points are first divided into a fixed quasi-random subset and an adaptive subset according to a specified fraction. The adaptive points are then further divided according to the two governing PDEs, allowing the method to incorporate both the enhanced domain coverage provided by the fixed quasi-random set and the resampling specificity.

For implementation, a first fraction determines the number of fixed quasi-random points. Following the initial training phase, this fraction of the total point budget is generated using a quasi-random algorithm and remains unchanged during subsequent resampling stages. The remaining points constitute the adaptive subsets, and their relative sizes are decided by a second fraction in a similar way. As in the PDE-partitioning strategy, separate probability distribution functions are obtained from each PDE. At every resampling stage, the two adaptive subsets are resampled from those. The union of the three subsets forms the full set of training points, with all carrying equal weight in the loss computation.

6.2.4 Resampling "Dropout"

A fourth strategy was explored on a smaller scale to assess the effect of retaining a random subset of previously used points at each resampling step. In this approach, a specified fraction determines how many points are excluded from resampling. These points retain their current locations, while the remaining points are replaced using the usual PDF-guided selection procedure (i.e. from Chapter 5). The points to

be excluded are randomly selected every resample, and so the method introduces a small degree of point 'persistence' across resampling steps, without permanently fixing any given collocation point.

For this investigation, no partitioning was applied, and residual information was used to guide resampling. Tests were carried out with $N = 20{,}000$ and $N = 50{,}000$. The dropout fraction was varied from 0.01 to 0.5, and the resulting performance was compared against the baseline case corresponding to a dropout fraction of 0.0, in which no points are retained and all are resampled at each step.

6.3 Results

In this results section, the investigations carried out to test the resampling and partitioning strategies described in the previous section are presented. In line with the aims of this chapter, the objective is to determine whether targeted resampling strategies can improve the efficiency or accuracy of resampling methods for systems of PDEs, especially as dimensionality increases. The need for such investigations arises from the limitations identified in the resampling strategy discussed in the previous chapter, particularly with respect to the attainable accuracy when solving coupled PDEs in three dimensions (two space plus time).

To address these aims, a sequence of experiments is carried out on the partitioning strategies introduced in Section 6.2, comparing the results directly against the modified resampling approach described in Section 5.3.3 using residual information as the guiding indicator. Each investigation focuses on how the allocation and resampling of collocation points influences the error as a function of the total number of points, and how that changes as the fraction of points in each partition changes.

The first investigation, presented in Section 6.3.1, assesses fixed-set partitioning, in which a proportion of collocation points is held constant in a quasi-random distribution while the remainder is resampled throughout training. First, this approach is tested for different guiding information sources to select one for following investigations. Then, different fractions of points in the fixed subset are tested. This is first done using a hyperparameter of $c = 1$ (Equation 2.1) in resampling as the usual default. This is then repeated with $c = 0$ to test the ability of the fixed-set partition to replace the need for the c hyperparameter.

Section 6.3.2 then examines PDE-based partitioning, where the collocation points are split between the individual PDE terms and resampled using information from its corresponding PDE residual. Similarly, guiding information is tested first, and then

the effect of varying the relative size of subsets. A combined three-way partitioning approach, bringing together the fixed-set and PDE-based strategies, is evaluated in Section 6.3.3. Finally, Section 6.3.4 presents a dropout-style experiment, in which a fixed fraction of points is deliberately excluded from resampling to reduce abrupt changes in point placement during training.

6.3.1 Fixed and Adaptive subsets

A comparison of the different guiding-information choices for resampling, at a fixed 50% split ratio and with hyperparameter $c = 0$ (as the fixed grid ensures coverage of the domain, so noise from c is not required), is shown in Figure 6.1. The figure presents how the error varies with the number of collocation points across the tested methods. In this figure, close agreement is observed between the no-partition baseline and partitioning using residual information for resampling with hyperparameter $k = 1$. While not improving over the baseline at this setting, it's significantly more effective than other derivatives and solution based information tested, and so is used for the following investigations.

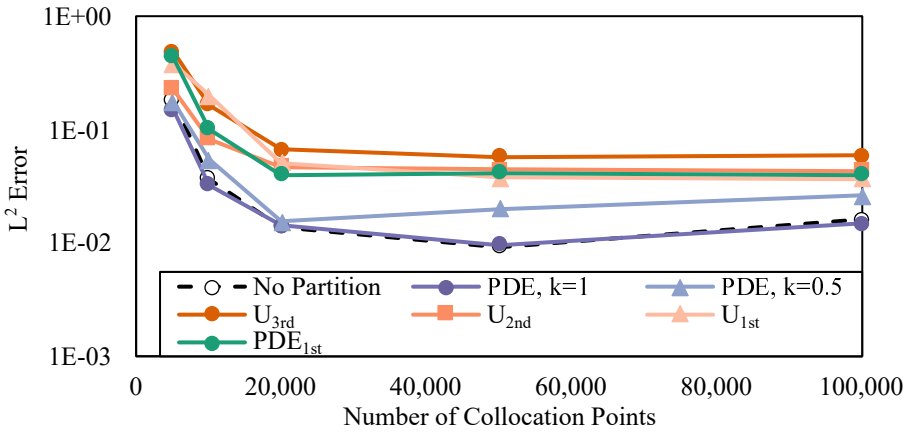

Figure 6.1: Error against number of collocation points using different guiding information for resampling, using hyperparameter $c = 0$, with 50% of points partitioned into a fixed set.

Figure 6.2 shows the variation in error with the number of collocation points for several choices of fixed-set fractions, using residual-based resampling. We first consider the case with hyperparameter $c = 1$, implementing noise in the adaptive set. Plotting against number of collocation points allows comparison against the case with no partition, and for values of $N \leq 50{,}000$, fractions of ≤ 0.25 again match the no partition baseline. A pattern of error increasing as N is further increased is observed similarly to the previous chapter. The exception to this is the case where the fraction $= 1$, where the entire grid is fixed and so a no-resampling method is

recovered. In this case, the it has the highest errors at low N, but at $N = 100{,}000$ matches the lowest errors obtained with an error of 9.12×10^{-3}.

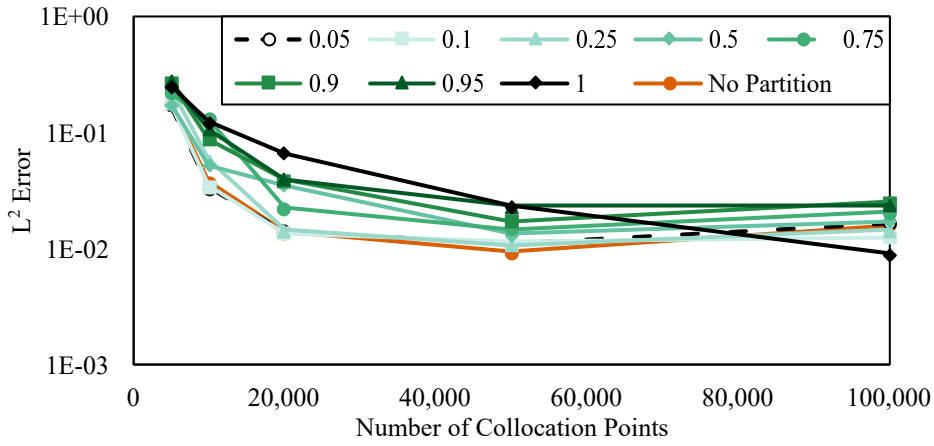

Figure 6.2: Error against number of collocation points, for varying fractions of points partitioned into the fixed set. Resampling uses PDE residual and hyperparameter $c = 1$.

To isolate the effect of the ratio, Figure 6.3 presents the same data but with error as a function of the fraction of points assigned to the fixed set, for several choices of total collocation points under the same resampling setup. The figure shows a few trends. First, we clearly see how error decreases steadily as the number of collocation points increases. The only exception to this is the $N = 100{,}000$ case, which is inferior except in the fraction=1 case, where we recover a no-resample strategy as all points are in the fixed set. We also observe that across $N \leq 50{,}000$, error slightly trends upwards as the fraction is increased.

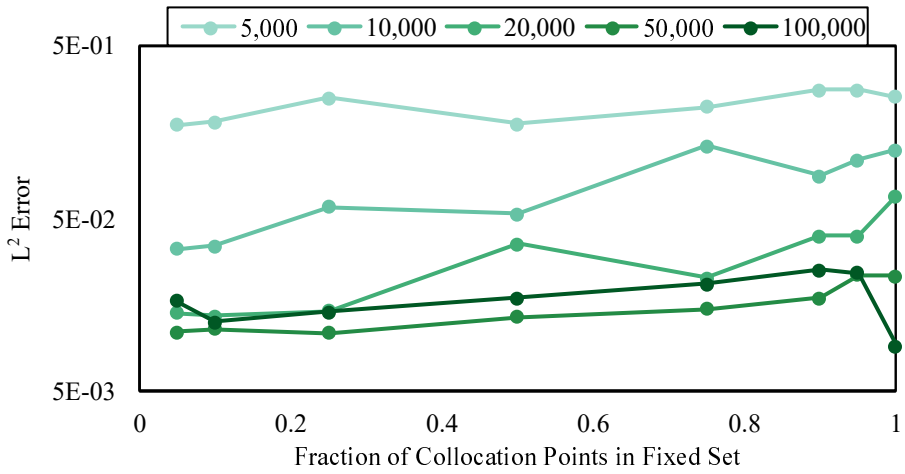

Figure 6.3: Error against fraction of points in fixed set, for varying number of collocation points. Resampling uses PDE residual and hyperparameter $c = 1$

Next we look at $c = 0$, removing the noise in the resampling set. This investigates whether the split partitioning freeing the resampling points to focus only on regions

of interest brings any benefits. Figure 6.4 shows how the error varies with the total number of collocation points for several choices of fixed-set fractions when $c = 0$. As before, this first plot provides a direct comparison against the baseline as the overall resolution increases. The dependence on the split ratio is then isolated in Figure 6.5, which displays the error as a function of the fraction of points assigned to the fixed set for a range of total collocation counts under the same $c = 0$ resampling procedure.

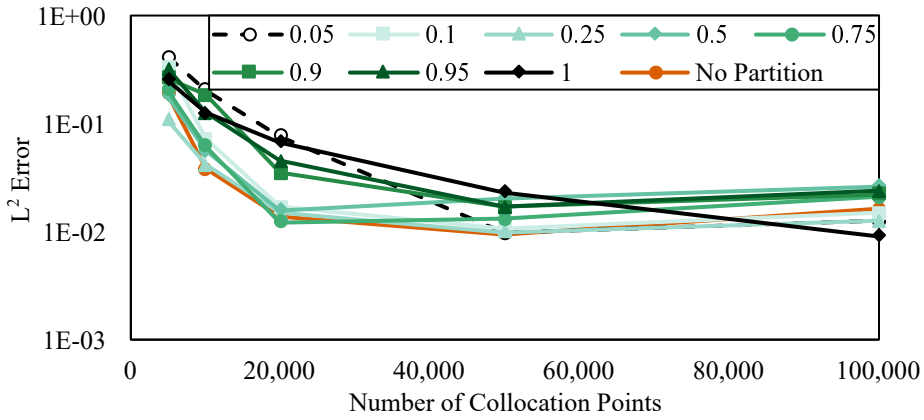

Figure 6.4: Error against number of collocation points, for varying fractions of points partitioned into the fixed set. Resampling uses PDE residual and hyperparameter $c = 0$.

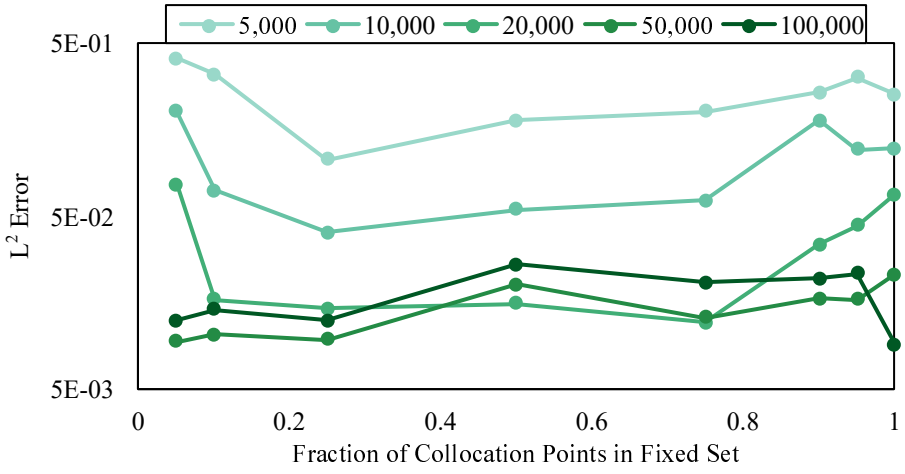

Figure 6.5: Error against fraction of points in fixed set, for varying number of collocation points. Resampling uses PDE residual and hyperparameter $c = 0$

For this $c = 0$ case, a number of features can be observed. As in the $c = 1$ case, the error is seen to decrease with increasing N, before rising again for $N = 100{,}000$, though several important differences are present. For $N \leq 20{,}000$, a sharp increase in error is observed as the ratio is reduced to 0.10 and 0.05. This behaviour no longer appears once N becomes sufficiently large, suggesting that a minimum number of

points may be required to maintain adequate coverage. The intermediate ratios between 0.10 and 0.75 exhibit similar levels of accuracy, with a general upward trend. Error increases more noticeably at ratios of 0.9 and 0.95, apart from the special case of ratio 1, where no points are resampled and the familiar pattern of steadily improving accuracy with increasing N is recovered. Overall, the best accuracies are found near ratios of 0.25 for $N = 20{,}000$, 50,000, and 100,000, taking values from 9.47×10^{-3} for $N = 50{,}000$ to 1.48×10^{-2} for $N = 20{,}000$. It is also noteworthy that for $N = 20{,}000$ the minimum error of 1.30×10^{-2} occurs at a ratio of 0.8, which is slightly lower than the 1.40×10^{-2} observed for the no-partition baseline, although the difference is not substantial.

Figure 6.6 presents the effect of the variant in which there is a translation of the quasi-random fixed set at each resampling step, comparing this approach against both the no-partition baseline and the fixed-set partitioning introduced above with a 0.5 ratio. This figure shows how the error varies with the number of collocation points across these different strategies. The translated fixed set, using $c = 0$, is observed to behave very similarly to the non-translated partitioned case with $c = 0$ and the no-partition baseline, with only slight improvements at $N = 10{,}000$ and $N = 20{,}000$, although the margins are small and may fall within expected variability (errors of 1.24% for the translated set compared with 1.42% and 1.39% for the no-partition baseline, representing a 13.5% central-difference change). At $N = 50{,}000$, the no-partition and non-translated variant with $c = 0$ yield the lowest errors, with 9.38×10^{-3}% and 9.73×10^{-3}%, respectively. By contrast, the $c = 1$ method performs worse across the range studied.

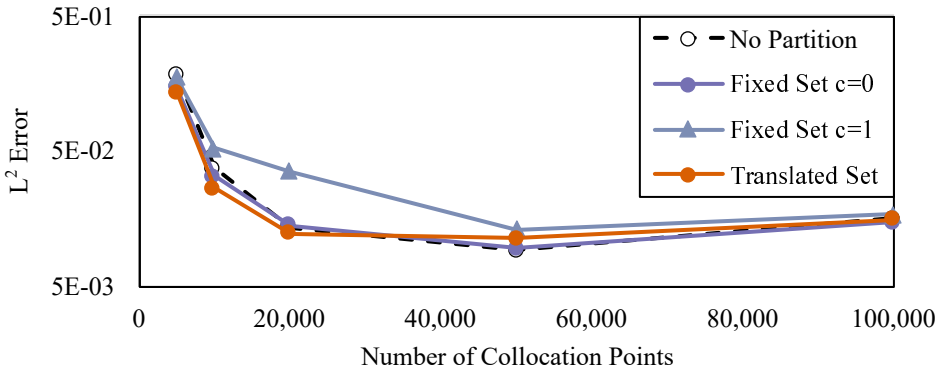

Figure 6.6: Error against number of collocation points, comparing a no partition approach (residual resampling and $c = 1$), a fixed set partition approach, and a translated quasi-random set. All partitioned methods use a fraction of 0.5 and residual resampling with $c = 0$.

6.3.2 PDE subsets

In this subsection, we consider a partitioning strategy in which the collocation points are split into two sets, each resampled using the information from a different PDE. The split is described by the fraction of points allocated to the first PDE's resampling set (which, in this case, corresponds to the PDE for the evolution of the u component of the solution), and we investigate the effect of varying this allocation. Our main focus is to determine whether resampling points according to a single PDE at a time can be more effective than using combined residual (or other) information. We vary the relative size of each set to account for possible differences in the importance of modelling each PDE, but the key question is whether any choice of allocation can achieve lower errors than a no-partition approach. The first figure compares different guiding-information choices to determine the method used in the subsequent investigations, and the following two figures present the behaviour against the total number of collocation points and then against the fraction assigned to the horizontal-PDE set (from here on we refer to the equation for the evolution of u as the horizontal PDE and the equation for the evolution of v as the vertical PDE).

The first investigation is given in Figure 6.7, and shows a comparison of guiding-information choices for the PDE-separated partitioning approach, using an equal 50% allocation of points to each PDE. This figure shows how the error varies with the number of collocation points for each method. Again using a pure residual resampling strategy is superior to curvature and higher derivatives, which have the same shape but plateau at errors of magnitude $\approx 4.5\% - 5.5\%$, compared to the minimum error of 1.04% for PDE at $N = 50{,}000$, or the baseline 0.94%. Across all displayed N, the residual-based method performs better than the alternatives, with no instance in which the derivative-based variants surpass it.

The effect of varying the total number of collocation points under the PDE-separated strategy is shown in Figure 6.8. Here, the error is plotted against the number of points for several fractions assigned to the horizontal-PDE set, using residual-based resampling with $c = 1$. This provides a direct comparison to the no-partition baseline. Figure 6.9 then isolates the dependence on the partition ratio by presenting the error as a function of the fraction of points allocated to the horizontal-PDE set, for several choices of total collocation counts under the same resampling setup.

Considering Figure 6.8, at low ratios we don't observe the re-increase in L^2

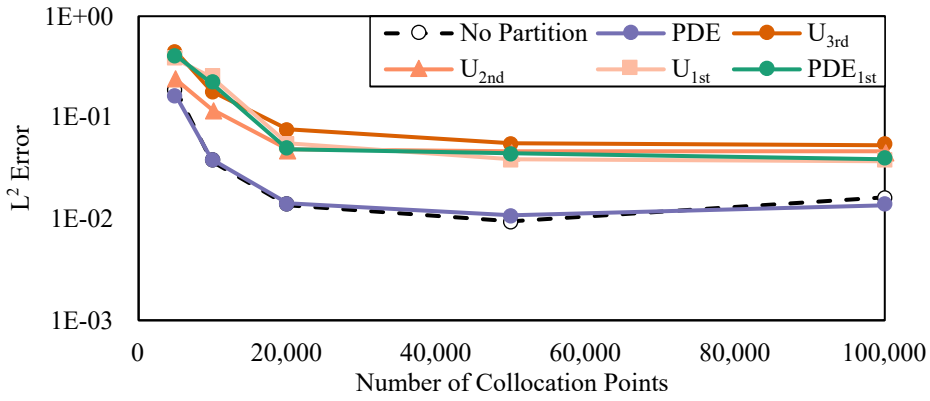

Figure 6.7: Error against number of collocation points using different guiding information for resampling, with an equal partition (50%) of points assigned to each PDE.

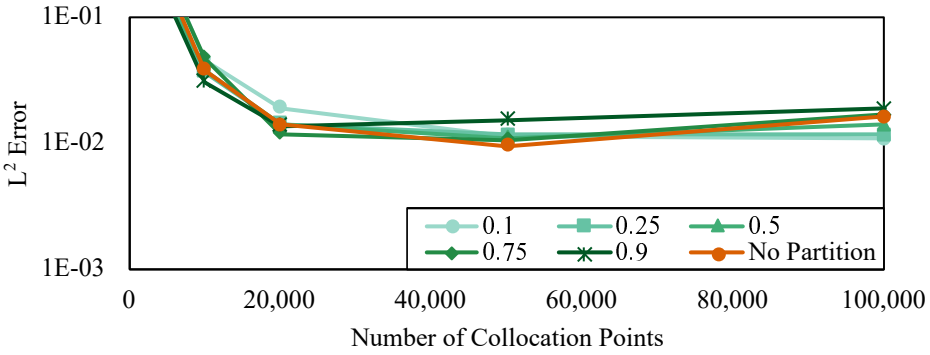

Figure 6.8: Error against number of collocation points, for varying fractions of collocation points assigned to the horizontal PDE partition. Resampling uses PDE residual and hyperparameter $c = 1$.

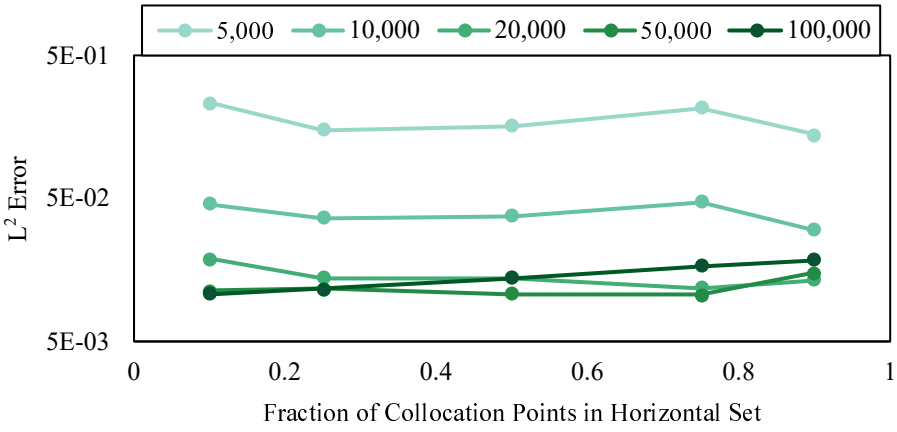

Figure 6.9: Error against fraction of collocation points assigned to the horizontal PDE partition, for varying number of collocation points. Resampling uses PDE residual and hyperparameter $c = 1$.

error, seeming instead to level off. However, these cases also require substantially more collocation points to attain the same accuracy as the higher ratios, suggesting that eventual deterioration may occur, only beyond the range of N tested. As previously noted in section 5.3.2, the observed reduction in accuracy at large N suggests that the overabundance of collocation points may increase the difficulty of training, outweighing the benefits from increased domain coverage. Considering next Figure 6.9, no clear overall pattern is visible for the smaller values of N, although at $N = 100{,}000$ a clearly linear increase in error with ratio can be observed. For $N = 20{,}000$ and $N = 50{,}000$, which typically yield the highest accuracy, the best results occur at a ratio of 0.8, with errors of 1.06% and 1.18%, representing only marginal improvements over the earlier methods tested.

The increase in error as the ratio is raised from 0.8 to 0.9 for these values of N suggests that a minimum number of points in the vertical set may be required for optimal training. For the lower N, the opposite behaviour is seen, with a decrease in error from 0.8 to 0.9, and the latter ratio giving the lowest error for those cases. At $N \leq 50{,}000$, a sharp decrease in error is observed when moving from a ratio of 0.1 to 0.2, indicating that a minimum number of points in the horizontal-PDE set may also be necessary. However, this does not appear to depend on the fraction itself but the number of points available, since the pattern disappears at $N = 50{,}000$ and $N = 100{,}000$, where enough points are available to cover the whole domain - and so benefits of a good resampling strategy are diluted. Overall, only limited variation is present between the intermediate ratio values, with the most pronounced changes occurring near the extremes.

6.3.3 Merged subset approach

In this final subsection, the two previous strategies are combined into a single three-way partitioning approach. The collocation points are divided equally between a fixed set, a resampling set targeting the horizontal component PDE, and a resampling set targeting the vertical component PDE. This allows assessment of whether the combined structure offers any advantage over the individual methods. As before, guiding-information choices are compared first. This is followed by an evaluation of the three-way partitioning against representative configurations from the earlier investigations to provide an overall comparison of all methods considered so far.

Figure 6.10 presents the comparison of guiding-information choices for the three-way partitioning strategy, using an equal allocation of points (one-third each) among the fixed set and the two PDE-specific sets. The figure shows how the error varies

with the number of collocation points relative to a no-partition baseline. As before, the PDE-residual-based method continues to yield the lowest overall error among the alternatives, outperforming both curvature and higher-derivative-based guidance. Even when the settings that performed best in the previous tests are applied here, no significant general advantage is observed. At $N = 10{,}000$, however, a modest improvement is seen, with an error of 2.57% compared with 3.75% for the no-partition baseline.

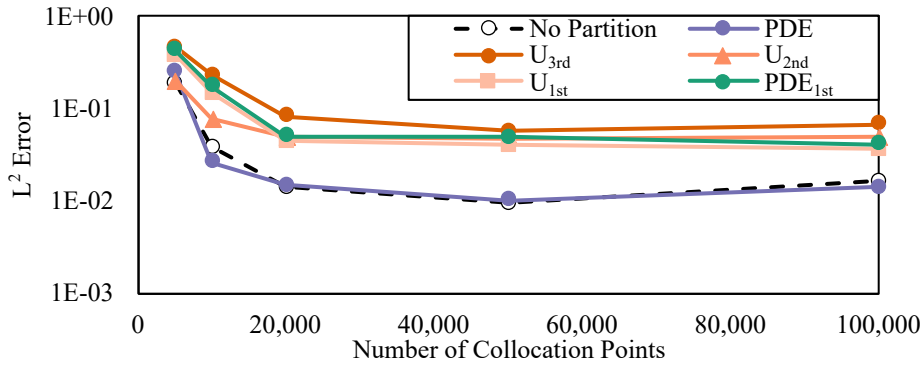

Figure 6.10: Error against number of collocation points using different guiding information for resampling, using $c = 0$, with points partitioned equally among a fixed set, the horizontal PDE, and the vertical PDE. This is compared to a no-partition approach.

Figure 6.11 then compares the three-way partitioning approach against representative settings from each of the earlier strategies, plotting the error against the total number of collocation points for all methods under residual-based resampling.Across the representative configurations shown in Figure 6.11, a very similar pattern is observed for all partitioning strategies, even when ratios are selected to give each method its best performance, reaching their lowest errors at $N = 50{,}000$ and then slightly deteriorating. At $N = 10{,}000$, the three-way partition and the translated fixed-set method exhibit slightly lower errors than the other approaches, with the improvement being a significant percentage (31.4% lower vs no-partition). By $N = 20{,}000$, however, the differences between methods narrow considerably, with the translated fixed-set method and PDE-split method performing best with 1.24% 1.18% error respectively, (14.91% lower vs no-partition). At $N = 50{,}000$, the lowest absolute errors - just below 1% - are achieved by the no-partition strategy, the fixed-set split, and the three-way partition, with the lowest being the no-partition strategy at 0.94%. Overall, no differences in behaviour are observed.

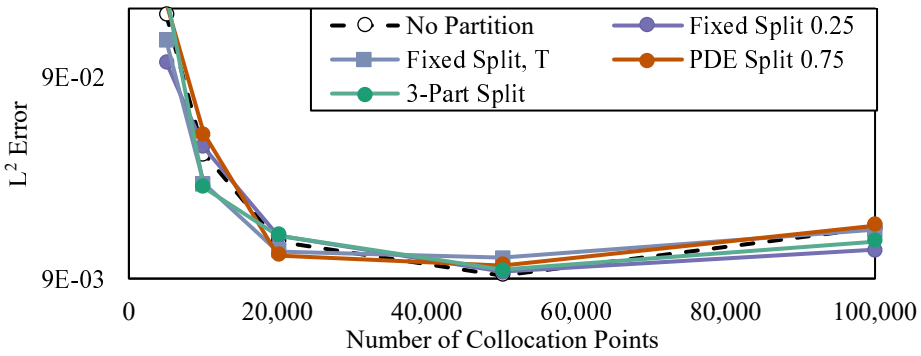

Figure 6.11: Error against total number of collocation points for representatives of each partitioning strategy: (i) no partition, (ii) fixed-set partition with ratio 0.25, (iii) fixed-set partition with translation and ratio 0.25 (labelled T), (iv) PDE-based partition with ratio 0.25, and (iv) three-part partition (1:1:1) between fixed-set and PDE sets. All using residual based resampling.

6.3.4 Dropout or Reservoir Sampling

A final investigation is carried out using the original resampling approach, but with a fraction of points randomly selected to remain unchanged during each resampling round. This setup is intended to test whether fixing part of the point set can mitigate issues arising from abrupt changes in the training set of collocation points.

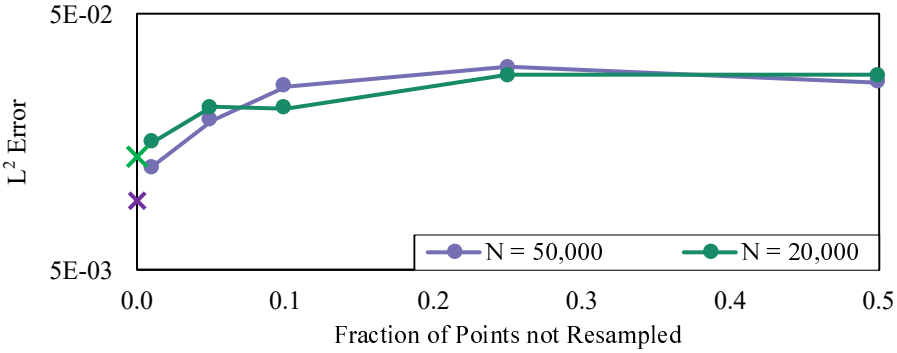

Figure 6.12: Error against fraction of points randomly selected to ignore resample, equivalent to the original approach at 0.0, for which the baseline error is shown. All obtained using residual based resampling, $c = 1$.

Figure 6.12 shows the resulting error as a function of the fraction of points excluded from resampling, where a value of 0.0 corresponds to the original method. All results are obtained using residual-based resampling with $c = 1$. As the fraction of points excluded from resampling increases, the final error becomes progressively worse, with the deterioration flattening out between ratios of approximately 0.3 and 0.5. Across the full range, the highest error is approximately 3.1%, compared with values below 1% for $N = 50{,}000$ and 1.39% for $N = 20{,}000$ without points fixing. As a result, no advantages of this exclusion strategy are observed for any of the cases

tested.

6.4 Discussion

6.4.1 Discussion of Results

Fixed-Partition

For the fixed-partition method, several features were observed across the various tests. Alternative guiding information, such as curvature or higher derivatives, consistently produced worse performance and were consequently not pursued in place of the residual. From the behaviour observed in Figure 6.6, it is reasonable to assert that the redundancy of having both a fixed grid and noisy resampling from $c = 1$ is detrimental, as it results in consistently higher error across the range of N, especially at $N = 20{,}000$.

With $c = 0$, (Figures 6.4, 6.5) the reduced accuracy at ratios of 0 and 0.05 suggests that a minimum number of points is required to maintain adequate coverage of the domain. Although splitting may provide a more resource-efficient allocation of points, it requires the number of points in each subset to be fixed in advance, which in practice would need to be estimated empirically, for example on the basis of domain volume. By contrast, selecting an appropriate value of c is less intuitive, as the resulting coverage cannot be quantified as easily and depends on the random behaviour of the candidate grid. The use of c does, however, introduce greater flexibility during training: as the residual decreases, the effective behaviour of the sampler shifts, with more points tending to resample randomly. Whether this dynamic is beneficial is not clear.

Focusing strictly on the improvements observed, the removal of c enables the lower ratios to achieve a much reduced error at $N = 20{,}000$ (Figure 6.4). Comparing to the no-partition baseline, reduced error is seen at $N < 50{,}000$, but the peak accuracy does not match the baseline across any ratio values tested. This suggests that the effect may be limited to shifting the required number of collocation points downward rather than improving the attainable accuracy. The recovery of the fixed-grid behaviour (taking up to $N = 100{,}000$ to match the minimum error achieved through adaptive resampling) at a fraction of 1 is a reminder of the benefits to of adaptive resampling. Namely, showing how adaptive resampling can often reach similar accuracies using fewer points, in this case approximately one-fifth. Although the computational cost is not strictly proportional to N (due to fixed overheads

in optimisation), in this setup the reduction in time taken from $N = 100{,}000$ to $N = 20{,}000$ would be of approximately 35%, from roughly 8 to 5 hours.

PDE Partition

For the PDE-based partition, the effect of splitting the collocation points across the separate PDEs was not found to be particularly significant. The fact that the best ratio was not 0.5 does lead us to believe that there is a difference in the relative importance of each PDE when it comes to training. However, what remains unclear is whether assigning points to each PDE with a ratio that is weighted to their relative importance is superior to an approach where the combined residual of both PDEs indirectly weights point selection. Again, a limitation of this implementation is that we also don't take into account whether the relative importance of each PDE changes throughout training, as our ratio is fixed. In contrast, a no-partition approach is unable to simultaneously target both PDEs if one has significantly higher residuals or derivatives as that will dominate the probability distribution function. However, this can also be interpreted positively: more collocation points are naturally allocated to the regions responsible for the largest errors, with no points ever being assigned to learn PDEs that have already been well optimised for .

Only slight patterns were observed at the extreme ratio values, suggesting that assigning a minimum number of points to each PDE may be optimum, but this value would be problem dependant and hard to estimate *a priori*. No increase in the maximum achievable accuracy was obtained through this strategy either, at any of the ratios, although the most accurate result at $N = 20{,}000$ was obtained with this strategy. From the results on final accuracy alone, it is thus hard to establish any conclusive answers to the question of whether targetting PDEs individually is overall better than adding all residual information together, as the results vary according to number of collocation points available. From a practical stand point, manual selection of ratios hinders the effectiveness of the approach, and a reasonable next step would be to implement the ratio selection as a learnable parameter, so that it may also adapt to the training environment.

Three-Part Split

The implementation of the combined strategy primarily served to demonstrate that the previously introduced approaches could be integrated straightforwardly and extended upon without substantial modification. The main limitation for the testing of this approach was the need for manual selection of subset sizes, as the require-

ment for two ratios to be specified greatly increased the number of combinations to test. As a result, only a representative case was looked at for equally-sized subsets. This choice was adequate for assessing the strategy, as based on the results of the individual partition strategies, as except for the extremes of subset partition, results were all close in accuracy to the baseline. The results showed this robustness, being the second best strategy at $N = 10{,}000$ and one of the three with accuracy $< 1\%$ at $N = 50{,}000$ (See Fig. 6.11).

Resample Dropout

Whilst the error seems to plateau in the range we investigate, we can infer what happens in the rest of the range thanks to the knowledge that at ratio=1 we would recover a fixed, no-resampling approach, as all points would ignore resampling. For that case, previous results obtained errors of 6.71% and 2.33% error (for $N = 20{,}000$ and $N = 50{,}000$ respectively). This confirms to us that it is unlikely that any amount of resampling dropout was effective for this problem.

Considering that the reason to test this implementation was to smoothen the transition from one training set of points to the next, we can conclude that this approach either does not help properly do this, or that perhaps the issue itself wasn't the transition of collocation points directly. There is still the option to looking into the optimisation process itself, to check whether learned weightings are being too drastically changed in between resamples, setting back the training process.

6.4.2 General Remarks

This chapter explored approaches that introduced small changes to resampling strategy and small increases in computational overhead but may increase accuracy of points, or reduce the number of collocation points required for a given accuracy. The application of these methods resulted in errors that matched the order of magnitude and changes with respect to number of points used as the baseline cases, but did not significantly improve on them. The best results were improvements of $\sim 30\%$ for the $N = 10{,}000$ case when using the three-part partition, and the fixed-set partition method with translation. While relatively significant improvements, these L_2 Errors of 2.57% are still over twice as large compared to the minimum possible errors achieved by the same model with $N = 20{,}000$.

While looking at these strategies as a proof of concept, we investigate a range of relative subset sizes (usually quantified as the fraction of total points) for our given problem, and these parameters are inherently very problem dependent. These

fractions are selected before training begins and then fixed, whereas in reality we have not been able to confirm whether the optimum ratio of points between subsets may change throughout training in response to optimisation. And while some of our observations allow us to begin to understand how certain fractions aid or hinder model training, in a practical application information such as the minimum number of points to assign to a given PDE or to ensure adequate coverage by a fixed subset of points will be unknown. For these reasons, the implementation of these ratios as learnable parameters that can be trained and adjusted throughout training is a necessary next step to better address our aims.

Something else to note about the manual selection of fractions is that due to the time and computational constraints of this thesis, very fine parameter searches weren't feasible. Especially in the case of ranges of N, more information may have been obtained from further investigations in the ranges between $N = 20{,}000$ and $N = 50{,}000$. This might have allowed us to find the absolute lowest error of each strategy and the number of points at which it was obtained, which may have served as a better comparison between methods.

Moving on from practical considerations, an aim was to see whether efficiency of collocation points can be improved by implementing additional sophistication to the methods. In particular, motivated from the results of the previous chapter, it was examined whether these strategies could achieve the levels of accuracy obtained for the one-dimensional Burgers' equation, or those achieved in the two-dimensional case at higher values of ν. With respect to these aims, the methods were successfully implemented, but errors of order 1×10^{-4} were not reproduced. Nevertheless, slight improvements were found at lower ranges of N for several of the strategies, and the analysis of the ratios used to divide fixed and adaptive subsets, and PDE subsets, provided further insight into the role of domain coverage and point distribution on training behaviour. From a broader perspective, our aim could be considered as identifying what the main limiting factor is preventing our network from training to the lower order of magnitude. Our observations have shown us that while our strategies have an effect and can help shift error and N required for a given accuracy down, something else may be limiting our PINN's maximum accuracy.

The dropout test served as a small scale investigation into a potential contributor to the erratic training patterns observed in Chapter 5 at low viscosity (Figure 5.18). It was hypothesised that his contributor could be the abrupt change in collocation points between optimisation steps due to resampling affecting stability. By ignoring resampling with a proportion of the total points, we hypothesised training might

be more smooth at the expense of some of the resampling benefits. However, the results indicated a clear negative effect on final error when implementing dropout, suggesting any stabilising influence was not sufficient to reduce the error, and so any benefits were outweighed by the loss of learning points. Although this constitutes a negative result, several avenues for further investigation remain. The findings indicate that the primary limitation may not lie in the point selection procedure itself, and further work focusing on the behaviour of the optimisation algorithm may prove to be more fruitful. Particularly, observing how weight updates respond to changing training data may provide insight into future improvements to adaptive resampling strategies.

Chapter 7

Conclusion

In this thesis physics-informed neural networks (PINNs), a developing method within machine learning for solving science and engineering problems, are examined. Their potential lies in applications where traditional numerical methods may not be the optimal choice, such as cases lacking full boundary condition information or where conventional solvers may be too expensive for the required level of accuracy. Because PINNs were a relatively new area of research at the project outset (known widely thanks to the paper Raissi et al. [2019]), this thesis's aims are exploratory in nature, established to evaluate PINNs' potential in more detail and to improve the more crude aspects of the methodology found in the contemporary literature.

For this end, PINNs are applied to simple fluid dynamic partial differential equations using a standard feedforward network architecture, and the influence of the choice and placement of collocation points on enforcing the underlying physics is investigated. In particular, random uniform sampling strategies are replaced by adaptive resampling methods. The thesis is guided by two general goals: first, to implement and refine adaptive resampling strategies in an effort to improve the sampling problem in PINNs; and second, to determine whether these adaptive approaches can enhance PINN performance in a universal and generalisable manner across different PDEs and problems.

In the subsections that follow, the objectives set within each chapter to achieve these general aims are outlined, and the accomplishments resulting from the work are summarised. Section 7.2 then discusses the outcomes in relation to the initial aims. The extent to which the contributions succeed within the broader context of the literature is then assessed, followed by a discussion of the limitations identified and possible directions for future work to address these.

7.1 Summary of findings

7.1.1 Chapter 4 - Aims and accomplishments

Following the preliminary work in Chapter 3, which established the benchmark and validated residual-based adaptivity from the literature, Chapter 4 is devoted to exploring alternative guiding indicators for carrying out adaptive resampling of collocation points. The first objective, inspired by traditional mesh refinement techniques in computational fluid dynamics, is to implement indicators derived from the solution itself such as velocity gradients and higher-order derivative information. The goal is to assess whether these indicators can reduce the number of collocation points needed to achieve a target accuracy or improve accuracy under a fixed computational budget, noting that the computational cost of PINN training increases approximately linearly with the number of collocation points, and thereby assessing whether they can outperform residual-based criteria.

A further aim is to test the generalisability of these approaches. To support this, the benchmark is extended by first comparing the resampling strategies on the one-dimensional Burgers' equation and then applying them to additional configurations and to another PDE, the Allen-Cahn equation. This enables an evaluation of how broadly these strategies perform beyond the initial test case.

Considering now the accomplishments, resampling strategies guided by residual derivatives and solution-based indicators are applied and compared against both existing adaptive methods and fixed random or pseudorandom point distributions. It is shown that adaptive point placement outperforms fixed distributions by achieving lower errors with fewer collocation points, with the mixed second derivative of the residual generally providing the strongest results among the different forms of guiding information.

For Burgers' equation, lower errors are achieved with fewer points than competing approaches, with performance depending on the complexity of the solution and on the chosen diffusion parameters. A threshold in the point count is identified, above which uniform Hammersley sampling becomes preferable, whilst adaptive methods consistently perform better below this threshold. Similar behaviour is confirmed for the Allen-Cahn equation, although the advantage over residual-only strategies is less significant.

The chapter also assesses several relevant factors that affect PINN performance, including the choice of optimisation algorithm, sensitivity to hyperparameters, and the handling of multiple indicators. Variations of these parameters are also tested,

for example an implementation that continuously adjusts the hyperparameter c throughout training. It is noted that although resampling introduces minimal computational overhead, PINNs remain expensive primarily because of the cost of the optimisation steps (which is proportional to the number of collocation points).

Overall, the chapter concludes that resampling strategies based on multiple indicators are viable, but that the interplay between sampling, optimisation, and PINN accuracy is complex and warrants further investigation, particularly when considering extensions to higher-dimensional problems.

7.1.2 Chapter 5 - Aims and accomplishments

Having established new methods for adaptive collocation point sampling with positive results in the previous chapter, work in Chapter 5 is directed towards addressing the goal of determining whether these findings generalise to more complex, and in particular high dimensional, settings. Therefore, the chapter focuses on extending the investigation from the one-dimensional setting to the unsteady two-dimensional Burgers' equations, formulated as a coupled system of two PDEs. The primary aim is to determine whether the benefits observed in the unsteady one-dimensional Burgers' Equation, such as improved accuracy and reduced collocation point count, are retained in higher dimensions and in systems of increasing complexity. A secondary aim is to examine the behaviour of solution-based guiding indicators in two dimensions and to consider the challenges and feasibility of employing higher-order derivatives as dimensionality increases.

To achieve these aims, a solution for the two-equation Burgers' system in 2D is generated and validated. A corresponding 2D PINN model is implemented, and the adaptive resampling strategies from earlier chapters are applied with adjustments to account for the increased dimensionality and second PDE. The discussion includes an examination of the memory load and scalability issues associated with the dense grid used in the resampling strategy. An iterative alternative is introduced to mitigate this memory burden, but the cost of computing higher derivatives as guiding indicators quickly scales with dimensionality with the current method.

The results show that adaptive resampling can still reduce the number of collocation points required to reach a given accuracy, particularly in the smoother $\nu = 0.01$ case. However, the training strategy of the adaptive method has to be modified significantly, and the network depth increased slightly compared with the 1D case in order to achieve stable training. Without these changes, noticeable training-stability issues are observed. In the $\nu = 0.01/\pi$ case, the lowest error achieved is markedly

higher than in the case with a smoother solution with $\nu = 0.01$.

The chapter provides a discussion of potential causes behind the training-stability issues encountered in the more complex problem, including the inability to consistently preserve progress between resampling steps and the potentially disruptive effect of completely replacing collocation point sets. Possible remedies suggested for further investigation include retaining subsets of points between resampling steps, employing more informative point-selection strategies, and adopting more robust optimisation schemes. Although the potential for adaptive resampling to reduce the collocation-point count is observed, the current implementation is found to be insufficient for scaling effectively to higher domain dimensionality. This outcome motivates the investigation into training robustness developed in the next chapter.

7.1.3 Chapter 6 - Aims and accomplishments

The aim of this chapter is to assess whether modifying how adaptive resampling is carried out - not only the frequency or quantity of resampling - can improve PINN accuracy or reduce the number of points required. Building on several limitations identified in the previous chapter, alternative ways of carrying out adaptive resampling are explored to address potential underlying causes. Specifically, three questions are considered: whether resampling only part of the collocation set, while keeping a fixed subset, can reduce variance between resamples and stabilise training; whether multiple-PDE systems benefit from equation-specific resampling (rather than combining residual information across equations); and whether a combined strategy incorporating both a fixed subset and per-equation adaptive subsets can balance domain coverage with focused learning. In all cases, parameter choices are examined to determine which are most effective for improving the performance on the two-dimensional problem with $\nu = 0.01/\pi$. Additionally, it is also aimed to investigate whether the sharp change in collocation-point locations during resampling might contribute to instability. This is done through a dropout-inspired modification that retains a random fraction of points at each resample to smooth loss transitions.

Systematic tests are then run across multiple values of N to evaluate how these strategies affect accuracy and scaling behaviour. The results show modest improvements - up to approximately 30% in some cases - relative to methods applied to the same problem, although none of the approaches reproduce the lower errors observed in the unsteady one-dimensional setting. The improvements are also found to be highly sensitive to the relative sizes chosen for the different subsets, with these ap-

pearing to be problem-dependent. This suggests making these parameters learnable may be necessary for practical application. The limited parameter search performed here prevents identification of optimal values of N or the minimum achievable errors for each strategy. Generally, the tested methods offer slight gains in accuracy, but are clearly not the limiting factor preventing the error from approaching the lower orders of error magnitude of $\approx 10^{-4}$ observed in Chapter 4.

A dropout-inspired approach is also implemented and is found to worsen accuracy. This indicates that the underlying issue is not caused simply by abrupt changes in collocation-point locations and that retaining points in fixed positions for longer hinders learning more than providing new locations at each resample. As a result, any stability improvements must be obtained by different strategies or through better implementations of adaptive resampling. Overall, it is concluded that refinements to resampling alone are insufficient, and that the observed instability may arise from optimisation behaviour rather than from the point-selection mechanism itself, which motivates a transition toward studying training-stability issues in future work.

7.2 Concluding remarks

7.2.1 Assessment of overall aims

The general aims of this thesis are twofold: first, to implement and refine adaptive resampling strategies in an effort to improve the sampling problem in PINNs; and second, to determine whether these adaptive approaches can enhance PINN performance in a universal and generalisable manner across different PDEs and problem settings.

Various strategies are implemented and refined, and the work in Chapter 4 demonstrates the benefits of using new sources of guiding information to guide the resampling for low-dimensional problems. This directly addresses the first aim and contributes to improving the sampling of collocation points in PINNs. Improvements of adaptive resampling over fixed resampling are first demonstrated, followed by additional gains obtained through the use of derivatives of residual and solution estimates. Adaptive resampling is observed to reduce computational cost by producing low errors utilising fewer points, with the overhead of the resampling process itself not being significant in comparison. Nonetheless, points for improvement are identified, as the probability distribution function method requires the selection of hyperparameters (that may be problem-dependant choices), and may

incur badly scaling memory costs. While these memory costs are manageable for the two-dimensional space-time domain, alternative approaches to resampling may be required to generalise to higher-volume domains, since constructing dense grids of candidate points becomes infeasible as domain volume increases exponentially with dimensionality.

These observations also relate to the second aim: testing the generalisability of the adaptive resampling strategies under consideration. This second aim is primarily tackled by the work in Chapters 5 and 6. In chapter 5, the strategies are extended to an additional spatial dimension, and the PINN is required to learn two PDEs simultaneously to model the system. The strategy is successfully adapted; however, even though improvements from adaptive resampling are still observed, these require modifications to the training procedure and result in significantly higher overall errors. Therefore, whilst adaptive resampling can still offer benefits in certain settings, the present methodology does not transfer straightforwardly, requiring non-trivial adjustments. Chapter 6 addresses this by attempting to improve on the methodology. The results show that the proposed alternatives do not significantly improve the generalisability. However, the resulting observations and discussions expand the understanding relevant to this aim, and may help inform future directions for improving the generalisability of adaptive resampling. In addition to the work in Chapters 5 and 6, which focuses on hyperbolic PDEs, a short investigation of a non-convex Poisson problem is presented in Appendix B, providing a further test of the methodology on an elliptical PDE.

7.2.2 Contributions to literature, limitations and future work.

Providing a brief review of the existing literature at the time the work in this thesis began is useful for situating the contributions of this thesis. Wu et al. [2023] presented the first synthesis of adaptive sampling methods for PINNs, building on earlier greedy algorithms such as those introduced in Lu et al. [2021b] to propose two adaptive sampling methods using a general Probability Distribution Function (PDF). Other methods developed during that period varied the PDF used to guide resampling (for example Tang et al. [2023]), otherwise modified the underlying training strategy [Subramanian et al., 2023].

Within this context, the work in this thesis has provided further evidence that adaptive resampling can optimise PINN training by increasing accuracy and reducing the number of collocation points required to produce a given accuracy. It further provides insight into how derivatives of residuals and of solution estimates can be

more effective indicators for selecting new points, and develops structured strategies utilising partitioning-based approaches to better distribute collocation points. Finally, the investigations performed on the two-dimensional coupled PDE benchmark provide some insight into the remaining challenges related to scalability and robustness of adaptive resampling as complexity increases. Nevertheless, this work has so far failed to deliver the robust and widely generalizable adaptive strategy that was the ultimate aim of this research.

Summarising the overall limitations identified in this work is also helpful for identifying directions for future work. One such limitation concerns the range of problem-dependent choices involved in PINN implementation, ranging from architecture selection to training strategy. In the context of adaptive sampling, this extends to questions such as when, and with what frequency, to resample the set of collocation points. These general limitations reflect the developing nature of the broader PINN methodology, and substantial research exists to address these general issues as discussed in Section 2.2.

A second limitation, more directly relevant to the adaptive resampling methodology involves the scalability of the implementation. Using indicator quantities based on gradients of the residual, the computation of first derivatives is relatively inexpensive (and often already required as part of the loss calculation), but the computation of higher-order derivatives becomes prohibitively expensive as the dimensionality of the domain increases. More importantly, the common approach of evaluating resampling indicators on a dense set of candidate points is highly inefficient and quickly becomes infeasible as domain size and dimension increases. This highlights the need for alternative mechanisms of obtaining the information needed for adaptive resampling.

It is also useful to consider work carried out in parallel with the writing of this thesis. Several similar PDF-based approaches have been developed, with work like Zhao and Ren [2025] merging resampling and weighting strategies to improve accuracy and robustness to noise. These PDF-based approaches nevertheless continue to rely on the evaluation of the PDF on a dense candidate set. Other recent work has sought to directly address this issue, for example by returning to cell-based or dynamically updated Voronoi discretisations. Methods of this type, such as Hu et al. [2026], avoid the scaling limitation by computing residuals at a small number of representative points that dictate collocation point distribution and further cell refinement. However, accuracy compared to other adaptive resampling methods may be limited due to points within each cell being sampled in a uniform random manner.

A different approach is presented by Visser et al. [2025] which aims to address scaling issues by repositioning collocation points individually through optimisation to maximise the squared residual. Their approach achieves comparable runtime and accuracy to standard formulations in one-dimensional cases, with an underlying strategy likely to be more scalable as the dimension of the domain increases. Another method that circumvents dense grid point evaluating is proposed by Zhang and Wang [2026], which improves performance on problems such as the Allen-Cahn Equation by formulating the sampling procedure as a multi-armed bandit problem. This balances the placement of points in high-residual areas with exploration of new areas, which optimises computational effort where residual is predicted to be most informative.

A further recent development of note by Zhang et al. [2025a] also extends beyond using residual information alone to guide resampling, similar to our work in Chapter 4). Their 'multi-benchmark' strategy considers the simultaneous use of a combination of network outputs (including their derivatives) together with components of the residual to instruct collocation point placement. This approach shows improved accuracy for coupled PDE systems, but the authors note the need for further development of 'benchmark' selection algorithms and new optimisation algorithms to further improve the application scope of PINNs.

Drawing on both the limitations identified in this thesis and from broader literature, several avenues for future work can be suggested. One broad avenue concerns the development of features that streamline the construction of tailored networks, for example through adaptive procedures that can learn appropriate hyperparameters during training where possible. Although this idea is not new, with examples including Liu et al. [2025]'s adaptive activation functions, further advancements in training strategy may be especially relevant for improving the applicability of resampling-based methods.

A second direction for further work, more directly related to the adaptive sampling techniques studied in this thesis, involves developing alternatives to the dense sets of candidate points for guiding resampling. As discussed earlier, several recent works have already begun exploring such alternatives, which is an important step for improving the scalability of adaptive resampling.

Finally, it would be valuable to investigate the underlying causes of the reduced accuracy observed in more complex two-dimensional problems. This includes analysing how network weights evolve as resampling progresses and identifying mechanisms that limit accuracy as problem dimensionality increases. Such investiga-

tions, together with improvements in accuracy and scalability offered by emerging approaches in the literature, may ultimately help improve the applicability and robustness of adaptive resampling strategies for more complex, real-world PDE systems.

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 3 2016. URL `http://arxiv.org/abs/1603.04467`.

Ahmed Al-Shafei, Hamidreza Zareipour, and Yankai Cao. High-Performance and Parallel Computing Techniques Review: Applications, Challenges and Potentials to Support Net-Zero Transition of Future Grids. *Energies*, 15(22):8668, 11 2022. ISSN 1996-1073. doi: 10.3390/en15228668.

Sokratis J. Anagnostopoulos, Juan Diego Toscano, Nikolaos Stergiopulos, and George Em Karniadakis. Residual-based attention and connection to information bottleneck theory in PINNs. 7 2023. URL `http://arxiv.org/abs/2307.00379`.

ANSYS Inc. *ANSYS Fluent Theory Guide, Section 24.1: Adaption Process*. Release 2024 r2 edition, 2024.

John Anticev, Ali Aghdaei, Wuxinlin Cheng, and Zhuo Feng. SGM-PINN: Sampling Graphical Models for Faster Training of Physics-Informed Neural Networks. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, New York, NY, USA, 6 2024. ACM. ISBN 9798400706011. doi: 10.1145/3649329.3656521.

Andreas C. Aristotelous, Edward C. Mitchell, and Vasileios Maroulas. ADLGM: An efficient adaptive sampling deep learning Galerkin method. *Journal of Com-*

putational Physics, 477:111944, 3 2023. ISSN 00219991. doi: 10.1016/j.jcp.2023.111944.

Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 11 2018. ISSN 09252312. doi: 10.1016/j.neucom.2018.06.056.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Steven L. Brunton. Applying machine learning to study fluid mechanics, 12 2021. ISSN 16143116.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. 5 2021. URL `http://arxiv.org/abs/2105.09506`.

Nanxi Chen, Sergio Lucarini, Rujin Ma, Airong Chen, and Chuanjie Cui. PF-PINNs: Physics-informed neural networks for solving coupled Allen-Cahn and Cahn-Hilliard phase field equations. *Journal of Computational Physics*, 529: 113843, 5 2025. ISSN 00219991. doi: 10.1016/j.jcp.2025.113843.

Elias Cueto and Francisco Chinesta. Thermodynamics of Learning Physical Phenomena. *Archives of Computational Methods in Engineering*, 30(8):4653–4666, 11 2023. ISSN 1134-3060. doi: 10.1007/s11831-023-09954-5.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next. *Journal of Scientific Computing*, 92(3):88, 9 2022. ISSN 0885-7474. doi: 10.1007/s10915-022-01939-z.

Arka Daw, Jie Bu, Sifan Wang, Paris Perdikaris, and Anuj Karpatne. Mitigating Propagation Failures in Physics-informed Neural Networks using Retain-Resample-Release (R3) Sampling. 6 2023.

M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 3 1994. ISSN 1069-8299. doi: 10.1002/cnm.1640100303.

Weinan E. The Dawning of a New Era in Applied Mathematics. *Notices of the American Mathematical Society*, 68(04):1, 4 2021. ISSN 0002-9920. doi: 10.1090/noti2259.

Weinan E, Jiequn Han, and Arnulf Jentzen. Algorithms for solving high dimensional PDEs: from nonlinear Monte Carlo to machine learning. *Nonlinearity*, 35(1):278–310, 1 2022. ISSN 0951-7715. doi: 10.1088/1361-6544/ac337f.

Abdelrahman Elmaradny, Ahmed Atallah, and Haithem Taha. Minimizing nature's cost: Exploring data-free physics-informed neural network solvers for fluid mechanics applications. *Physics of Fluids*, 37(2), 2 2025. ISSN 1070-6631. doi: 10.1063/5.0250022.

Amer Farea, Olli Yli-Harja, and Frank Emmert-Streib. Understanding Physics-Informed Neural Networks: Techniques, Applications, Trends, and Challenges. *AI*, 5(3):1534–1557, 8 2024. ISSN 2673-2688. doi: 10.3390/ai5030074.

Wentao Feng, Ming Hu, Rui Huang, Chenwei Tang, Shudong Huang, and Jiancheng Lv. DSAIS-PINN: Dynamic seeds allocation importance sampling for physics-informed neural networks. *Neurocomputing*, 648:130578, 10 2025. ISSN 09252312. doi: 10.1016/j.neucom.2025.130578.

Jose Florido, He Wang, Amirul Khan, and Peter K. Jimack. Investigating Guiding Information for Adaptive Collocation Point Sampling in PINNs. pages 323–337. 2024. doi: 10.1007/978-3-031-63759-9{_}36.

Roy Frostig, Matthew James Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. In *SysML conference 2018*, 2019.

Qinjiao Gao, Zuowei Wang, Ran Zhang, and Dongjiang Wang. Adaptive Movement Sampling Physics-Informed Residual Network (AM-PIRN) for Solving Nonlinear Option Pricing models. 4 2025.

Zhiwei Gao, Liang Yan, and Tao Zhou. Failure-Informed Adaptive Sampling for PINNs. *SIAM Journal on Scientific Computing*, 45(4):A1971–A1994, 2023. doi: https://doi.org/10.1137/22M1527763.

Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403, 2 2020. ISSN 10902716. doi: 10.1016/j.jcp.2019.109056.

Ian. Goodfellow, Yoshua. Bengio, and Aaron. Courville. *Deep learning*. The MIT Press, 2017. ISBN 9780262035613.

Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-Informed Deep Neural Operator Networks. 7 2022. URL http://arxiv.org/abs/2207.05748.

Yiqi Gu, Haizhao Yang, and Chao Zhou. SelectNet: Self-paced learning for high-dimensional partial differential equations. *Journal of Computational Physics*, 441: 110444, 9 2021. ISSN 00219991. doi: 10.1016/j.jcp.2021.110444.

J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2(1):84–90, 12 1960. ISSN 0029-599X. doi: 10.1007/BF01386213.

David A. Ham, Paul H. J. Kelly, Lawrence Mitchell, Colin J. Cotter, Robert C. Kirby, Koki Sagiyama, Nacime Bouziani, Sophia Vorderwuelbecke, Thomas J. Gregory, Jack Betteridge, Daniel R. Shapero, Reuben W. Nixon-Hill, Connor J. Ward, Patrick E. Farrell, Pablo D. Brubeck, India Marsden, Thomas H. Gibson, Miklós Homolya, Tianjiao Sun, Andrew T. T. McRae, Fabio Luporini, Alastair Gregory, Michael Lange, Simon W. Funke, Florian Rathgeber, Gheorghe-Teodor Bercea, and Graham R. Markall. *Firedrake User Manual*. Imperial College London and University of Oxford and Baylor University and University of Washington, first edition edition, 5 2023.

J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Springer Netherlands, Dordrecht, 1964. ISBN 978-94-009-5821-0. doi: 10.1007/978-94-009-5819-7.

Jiequn Han, Arnulf Jentzen, and E. Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences of the United States of America*, 115(34):8505–8510, 8 2018. ISSN 10916490. doi: 10.1073/pnas.1718942115.

John M Hanna, José V Aguado, Sebastien Comas-Cardona, Ramzi Askri, and Domenico Borzacchiello. Residual-based adaptivity for two-phase flow simulation in porous media using Physics-informed Neural Networks. *Computer Methods in Applied Mechanics and Engineering*, 396:115100, 2022. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2022.115100.

Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Max Rietmann, Jose del Aguila Ferrandis, Won-

min Byeon, Zhiwei Fang, and Sanjay Choudhry. NVIDIA SimNetˆ{TM}: an AI-accelerated multi-physics simulation framework. 12 2020. URL http://arxiv.org/abs/2012.07938.

J. J. Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535):33–36, 7 1995. ISSN 0028-0836. doi: 10.1038/376033a0.

Jie Hou, Ying Li, and Shihui Ying. Enhancing PINNs for solving PDEs via adaptive collocation point movement and adaptive loss weighting. *Nonlinear Dynamics*, 111 (16):15233–15261, 8 2023. ISSN 0924-090X. doi: 10.1007/s11071-023-08654-w.

Weifei Hu, Jiale Liao, Jiquan Yan, Jianhao Fang, Feng Zhao, Ikjin Lee, and Jianrong Tan. Spatiotemporal-dependent reliability analysis with adaptive sampling physics-informed neural networks. *Reliability Engineering & System Safety*, 265: 111560, 1 2026. ISSN 09518320. doi: 10.1016/j.ress.2025.111560.

Zheyuan Hu, Ameya D. Jagtap, George Em Karniadakis, and Kenji Kawaguchi. Augmented Physics-Informed Neural Networks (APINNs): A gating network-based soft domain decomposition methodology. *Engineering Applications of Artificial Intelligence*, 126:107183, 11 2023. ISSN 09521976. doi: 10.1016/j.engappai.2023.107183.

Ameya D Jagtap and George Em Karniadakis. Extended physics-informed neural networks (XPINNs) : A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5):2002–2041, 2020.

Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 3 2020a. ISSN 00219991. doi: 10.1016/j.jcp.2019.109136.

Ameya D. Jagtap, Ehsan Kharazmi, and George Em Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365, 6 2020b. ISSN 00457825. doi: 10.1016/j.cma.2020.113028.

Hrvoje Jasak, Aleksandar Jemcov, and Željko Tuković. OpenFOAM: A C++ Library for Complex Physics Simulations. In *International*

Workshop on Coupled Methods in Numerical Dynamics, 2007. URL https://api.semanticscholar.org/CorpusID:35226827.

Zhengrong Jin, Wanrong Zan, Shichao Ma, and Wantao Jia. Physics-Informed Neural Networks with hybrid sampling for stationary Fokker–Planck–Kolmogorov Equation. *Physica A: Statistical Mechanics and its Applications*, 663:130434, 4 2025. ISSN 03784371. doi: 10.1016/j.physa.2025.130434.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning, 6 2021. ISSN 25225820.

Mahmoud Khadijeh, Veronica Cerqueglini, Cor Kasbergen, Sandra Erkens, and Aikaterini Varveri. Multistage physics informed neural network for solving coupled multiphysics problems in material degradation and fluid dynamics. *Engineering with Computers*, 6 2025. ISSN 0177-0667. doi: 10.1007/s00366-025-02174-4.

Ehsan Kharazmi, Zhongqiang Zhang, and George E.M. Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2 2021. ISSN 00457825. doi: 10.1016/j.cma.2020.113547.

Aditi S. Krishnapriyan, Amir Gholami, Shandian Zhe, Robert M. Kirby, and Michael W. Mahoney. Characterizing possible failure modes in physics-informed neural networks. 9 2021. URL http://arxiv.org/abs/2109.01050.

Isaac Elias Lagaris, Aristidis Likas, and Dimitrios I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, 1998. ISSN 10459227. doi: 10.1109/72.712178.

Gregory Kang Ruey Lau, Apivich Hemachandra, See-Kiong Ng, and Bryan Kian Hsiang Low. PINNACLE: PINN Adaptive ColLocation and Experimental points selection. 4 2024. URL http://arxiv.org/abs/2404.07662.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 12 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541.

Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient Based Learning applied to Document Recognition. In *Proceedings of the IEEE*, 1998. doi: 10.1109/5.726791.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436–444, 5 2015. ISSN 0028-0836. doi: 10.1038/nature14539.

Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3M: A Deep Domain Decomposition Method for Partial Differential Equations. *IEEE Access*, 8:5283–5294, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2957200.

Wensheng Li, Chao Zhang, Chuncheng Wang, Hanting Guan, and Dacheng Tao. Revisiting PINNs: Generative Adversarial Physics-informed Neural Networks and Point-weighting Method, 8 2022.

Wensheng Li, Chuncheng Wang, Hanting Guan, Jian Wang, Jie Yang, Chao Zhang, and Dacheng Tao. Generative adversarial physics-informed neural networks for solving forward and inverse problem with small labeled samples. *Computers & Mathematics with Applications*, 183:98–120, 4 2025. ISSN 08981221. doi: 10.1016/j.camwa.2025.01.025.

Xin Yang Liu and Jian Xun Wang. Physics-informed Dyna-style model-based deep reinforcement learning for dynamic control. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2255), 2021. ISSN 14712946. doi: 10.1098/rspa.2021.0618.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov-Arnold Networks. 2 2025.

Zichao Long, Yiping Lu, and Bin Dong. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 12 2019. ISSN 00219991. doi: 10.1016/j.jcp.2019.108925.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 3 2021a. ISSN 25225839. doi: 10.1038/s42256-021-00302-5.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021b. ISSN 00361445. doi: 10.1137/19M1274067.

Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-Informed Neural Networks with Hard Constraints for

Inverse Design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 1 2021c. ISSN 1064-8275. doi: 10.1137/21M1397908.

Marlon S. Mathias, Wesley P. de Almeida, Jefferson F. Coelho, Lucas P. de Freitas, Fellipe M. Moreno, Caio F. D. Netto, Fabio G. Cozman, Anna Helena Reali Costa, Eduardo A. Tannuri, Edson S. Gomi, and Marcelo Dottori. Augmenting a Physics-Informed Neural Network for the 2D Burgers Equation by Addition of Solution Data Points. In João Carlos Xavier-Junior and Ricardo Araujo Rios, editors, *Intelligent Systems*, pages 388–401, Cham, 2022. Springer International Publishing. ISBN 978-3-031-21689-3.

Levi D. McClenny and Ulisses M. Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474, 2 2023. ISSN 10902716. doi: 10.1016/j.jcp.2022.111722.

M. D. McKay, R. J. Beckman, and W. J. Conover. Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239–245, 5 1979. ISSN 0040-1706. doi: 10.1080/00401706.1979.10489755.

Anthony F Molland and Stephen R Turnock. Theoretical and numerical methods. In *Marine Rudders and Control Surfaces*, pages 233–311. Elsevier, 2007. doi: 10.1016/B978-075066944-3/50009-3.

Mohammad Amin Nabian, Rini Jasmine Gladstone, and Hadi Meidani. Efficient training of physics-informed neural networks via importance sampling. *Computer-Aided Civil and Infrastructure Engineering*, 36(8):962–977, 8 2021. ISSN 1093-9687. doi: 10.1111/mice.12685.

Nils J. Nilsson. *Learning machines : foundations of trainable pattern-classifying systems*. McGraw-Hill, New York, 1965.

Houman Owhadi. Bayesian Numerical Homogenization. *Multiscale Modeling & Simulation*, 13(3):812–828, 1 2015. ISSN 1540-3459. doi: 10.1137/140974596.

Guofei Pang, Lu Lu, and George Em Karniadakis. fPINNs: Fractional Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 1 2019. ISSN 1064-8275. doi: 10.1137/18M1229845.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban

Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Michael Penwarden, Ameya D. Jagtap, Shandian Zhe, George Em Karniadakis, and Robert M. Kirby. A unified scalable framework for causal sweeping strategies for Physics-Informed Neural Networks (PINNs) and their temporal decompositions. 2 2023. URL `http://arxiv.org/abs/2302.14227`.

Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Réglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, 4 2019. ISSN 0022-1120. doi: 10.1017/jfm.2019.62.

Christopher Rackauckas and Qing Nie. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *Journal of Open Research Software*, 5(1):15, 5 2017. doi: 10.5334/jors.151.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378: 686–707, 2 2019. ISSN 10902716. doi: 10.1016/j.jcp.2018.10.045.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 3 2018. ISSN 00219991. doi: 10.1016/j.jcp.2017.11.039.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Inferring solutions of differential equations using noisy multi-fidelity data. *Journal of Computational Physics*, 335:736–746, 4 2017a. ISSN 00219991. doi: 10.1016/j.jcp.2017.01.060.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics*, 348:683–693, 11 2017b. ISSN 00219991. doi: 10.1016/j.jcp.2017.07.050.

Ibai Ramirez, Joel Pino, David Pardo, Mikel Sanz, Luis del Rio, Alvaro Ortiz, Kateryna Morozovska, and Jose I. Aizpurua. Residual-based attention Physics-informed Neural Networks for spatio-temporal ageing assessment of transformers

operated in renewable power plants. *Engineering Applications of Artificial Intelligence*, 139:109556, 1 2025. ISSN 09521976. doi: 10.1016/j.engappai.2024.109556.

Spyros Rigas, Michalis Papachristou, Theofilos Papadopoulos, Fotios Anagnostopoulos, and Georgios Alexandridis. Adaptive Training of Grid-Dependent Physics-Informed Kolmogorov-Arnold Networks. *IEEE Access*, 12:176982–176998, 2024. ISSN 2169-3536. doi: 10.1109/ACCESS.2024.3504962.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 10 1986. ISSN 0028-0836. doi: 10.1038/323533a0.

Pushan Sharma, Wai Tong Chung, Bassem Akoush, and Matthias Ihme. A Review of Physics-Informed Machine Learning in Fluid Mechanics, 3 2023. ISSN 19961073.

Fergus Shone, Nishant Ravikumar, Toni Lassila, Michael MacRaild, Yongxing Wang, Zeike A. Taylor, Peter Jimack, Erica Dall'Armellina, and Alejandro F. Frangi. Deep Physics-Informed Super-Resolution of Cardiac 4D-Flow MRI. In *International conference on information processing in medical imaging*, pages 511–522. Springer, 2023. doi: 10.1007/978-3-031-34048-2{_}39.

Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375: 1339–1364, 12 2018. ISSN 10902716. doi: 10.1016/j.jcp.2018.08.029.

I.M Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4): 86–112, 1 1967. ISSN 00415553. doi: 10.1016/0041-5553(67)90144-9.

Hwijae Son, Sung Woong Cho, and Hyung Ju Hwang. Enhanced physics-informed neural networks with Augmented Lagrangian relaxation method (AL-PINNs). *Neurocomputing*, 548:126424, 9 2023. ISSN 09252312. doi: 10.1016/j.neucom.2023.126424.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435.

Michael Stein. Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics*, 29(2):143–151, 5 1987. ISSN 0040-1706. doi: 10.1080/00401706.1987.10488205.

Shashank Subramanian, Robert M. Kirby, Michael W. Mahoney, and Amir Gholami. Adaptive Self-Supervision Algorithms for Physics-Informed Neural Networks. In *Frontiers in Artificial Intelligence and Applications*, volume 372, pages 2234–2241. IOS Press BV, 9 2023. ISBN 9781643684369. doi: 10.3233/FAIA230521.

N. Sukumar and Ankit Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2 2022. ISSN 00457825. doi: 10.1016/j.cma.2021.114333.

Amirtahà Taebi. Deep Learning for Computational Hemodynamics: A Brief Review of Recent Advances. *Fluids*, 7(6):197, 6 2022. doi: 10.3390/fluids7060197.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020.

Keju Tang, Xiaoliang Wan, and Qifeng Liao. Deep density estimation via invertible block-triangular mapping. *Theoretical and Applied Mechanics Letters*, 10(3):143–148, 3 2020. ISSN 20950349. doi: 10.1016/j.taml.2020.01.023.

Kejun Tang, Xiaoliang Wan, and Chao Yang. DAS-PINNs: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, 3 2023. ISSN 00219991. doi: 10.1016/j.jcp.2022.111868.

Homayoon Tarbiyati and Behzad Nemati Saray. Weight initialization algorithm for physics-informed neural networks using finite differences. *Engineering with Computers*, 40(3):1603–1619, 6 2024. ISSN 0177-0667. doi: 10.1007/s00366-023-01883-y.

Jorge F. Urbán, Petros Stefanou, and José A. Pons. Unveiling the optimization process of physics informed neural networks: How accurate and competitive can PINNs be? *Journal of Computational Physics*, 523:113656, 2 2025. ISSN 00219991. doi: 10.1016/j.jcp.2024.113656.

Ricardo Vinuesa and Steven L. Brunton. Enhancing Computational Fluid Dynam-

ics with Machine Learning. 10 2021. doi: 10.1038/s43588-022-00264-7. URL http://arxiv.org/abs/2110.02085http://dx.doi.org/10.1038/s43588-022-00264-7.

Coen Visser, Alexander Heinlein, and Bianca Giovanardi. Pacmann: Point adaptive collocation method for artificial neural networks, 2025. URL https://arxiv.org/abs/2411.19632.

Xiaoliang Wan, Tao Zhou, and Yuancheng Zhou. Adaptive Importance Sampling for Deep Ritz. *Communications on Applied Mathematics and Computation*, 7(3): 929–953, 6 2025. ISSN 2096-6385. doi: 10.1007/s42967-024-00422-w.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 1 2021. ISSN 1064-8275. doi: 10.1137/20M1318043.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110768. URL https://www.sciencedirect.com/science/article/pii/S002199912100663X.

Sifan Wang, Shyam Sankaran, Hanwen Wang, and Paris Perdikaris. An Expert's Guide to Training Physics-informed Neural Networks. 8 2023a.

Y. Wang, C. Y. Lai, J. Gómez-Serrano, and T. Buckmaster. Asymptotic Self-Similar Blow-Up Profile for Three-Dimensional Axisymmetric Euler Equations Using Neural Networks. *Physical Review Letters*, 130(24), 6 2023b. ISSN 10797114. doi: 10.1103/PhysRevLett.130.244002.

Yanjie Wang, Feng Liu, Faguo Wu, and Xiao Zhang. Navigating PINNs via maximum residual-based continuous distribution. *Communications in Nonlinear Science and Numerical Simulation*, 141:108460, 2 2025a. ISSN 10075704. doi: 10.1016/j.cnsns.2024.108460.

Ziwen Wang, Qimin Wang, Weisheng Zhang, Sheng Zhang, Shanmei Chen, and Chao Li. Enhanced Physics-Informed Neural Network with Coarse Mesh Finite Element Pretraining. In *2025 5th International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, pages 1525–1528. IEEE, 1 2025b. ISBN 979-8-3315-0796-1. doi: 10.1109/NNICE64954.2025.11063871.

Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, and Lu Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed

neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403, 1 2023. ISSN 00457825. doi: 10.1016/j.cma.2022.115671.

Zixue Xiang, Wei Peng, Xu Liu, and Wen Yao. Self-adaptive loss balanced Physics-informed neural networks. *Neurocomputing*, 496:11–34, 7 2022. ISSN 09252312. doi: 10.1016/j.neucom.2022.05.015.

Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *Journal of Computational Physics*, 425:109913, 1 2021. ISSN 00219991. doi: 10.1016/j.jcp.2020.109913.

Zijiang Yang, Zhongwei Qiu, and Dongmei Fu. DMIS: Dynamic Mesh-Based Importance Sampling for Training Physics-Informed Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4):5375–5383, 6 2023. ISSN 2374-3468. doi: 10.1609/aaai.v37i4.25669.

Jeremy Yu, Lu Lu, Xuhui Meng, and George Em Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems. *Computer Methods in Applied Mechanics and Engineering*, 393, 4 2022. ISSN 00457825. doi: 10.1016/j.cma.2022.114823.

Yaohua Zang, Gang Bao, Xiaojing Ye, and Haomin Zhou. Weak adversarial networks for high-dimensional partial differential equations. *Journal of Computational Physics*, 411:109409, 6 2020. ISSN 00219991. doi: 10.1016/j.jcp.2020.109409.

Bastian Zapf, Johannes Haubner, Miroslav Kuchta, Geir Ringstad, Per Kristian Eide, and Kent-Andre Mardal. Investigating molecular transport in the human brain from MRI with physics-informed neural networks. *Scientific Reports*, 12(1): 15475, 9 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-19157-w.

Shaojie Zeng, Zong Zhang, and Qingsong Zou. Adaptive deep neural networks methods for high-dimensional partial differential equations. *Journal of Computational Physics*, 463:111232, 8 2022. ISSN 00219991. doi: 10.1016/j.jcp.2022.111232.

Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 11 2019. ISSN 00219991. doi: 10.1016/j.jcp.2019.07.048.

Fangyi Zhang and Dianpeng Wang. TSPINN: Thompson sampling-based adaptive training for physics-informed neural networks. *Statistics and Computing*, 36(2): 75, 4 2026. ISSN 0960-3174. doi: 10.1007/s11222-026-10824-w.

Guangtao Zhang, Huiyu Yang, Fang Zhu, Yang Chen, and xiaoning zheng. Dasa-Pinns: Differentiable Adversarial Self-Adaptive Pointwise Weighting Scheme for Physics-Informed Neural Networks. *SSRN Electronic Journal*, 8 2023. doi: 10.2139/ssrn.4376049.

Yabin Zhang, Liang-Jian Deng, and Minyu Feng. Multi-benchmark adaptive sampling physics-informed neural network for complex and coupled equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 35(10), 10 2025a. ISSN 1054-1500. doi: 10.1063/5.0296969.

Zhaoyang Zhang, Qingwang Wang, Yinxing Zhang, Tao Shen, and Weiyi Zhang. Physics-informed neural networks with hybrid Kolmogorov-Arnold network and augmented Lagrangian function for solving partial differential equations. *Scientific Reports*, 15(1):10523, 3 2025b. ISSN 2045-2322. doi: 10.1038/s41598-025-92900-1.

Zhao Zhao and Bo Ren. PINNs with hybrid residual-driven adaptive sampling and weighted loss for the (2+1)-dimensional gpKP equation. *Physics Letters A*, 562: 130998, 12 2025. ISSN 03759601. doi: 10.1016/j.physleta.2025.130998.

Wen Zhou, Shuichiro Miwa, and Koji Okamoto. Advancing fluid dynamics simulations: A comprehensive approach to optimizing physics-informed neural networks. *Physics of Fluids*, 36(1), 1 2024. ISSN 10897666. doi: 10.1063/5.0180770.

Appendix A

Additional Figures

A.1 Visualisation of collocation point distribution

As the adaptive resampling methods employed in this thesis change the set of collocation points multiple times throughout training, it can be difficult to visualise the overall sampling behaviour of each method from observing individual sets of points, especially as guiding indicators derived from the loss may focus different areas as training advances and losses are reduced. However, by plotting all points placed throughout the training process, a 2D histogram can be constructed to observe the broad patterns in the behaviour of the three main methods employed in Chapter 4. This histogram is shown in Figure A.1 below.

Side by side, we can observe that adaptive resampling guided by the PDE residual is the most subtle of the methods, only slightly increasing the bias towards the middle of the domain. Using U_{xt}, we can see that points are more likely to be along $x = 0$ with very little deviation, especially concentrating around values of t near the formation of the 'shock'. The PDE_{xt} method is biased towards the middle region but is not as constrained as U_{xt} to the exact location of the shock, sampling between $x = \pm 0.1$, similarly showing a slightly greater concentration at earlier t, around $t \approx 0.4$.

In order to accompany the solutions from Figures 5.7 and 5.8, using residual-based adaptive resampling, a similar visualisation is constructed for the 2D Burgers' equation. As collocation points are distributed along the three-dimensional domain (x, y, t), the temporal domain is split into four equal intervals, with the corresponding spatial distributions presented as four separate histograms in Figure A.2.

For this case, we can observe that the locations where most points are concentrated align closely with the regions of high error shown in Figures 5.7 and 5.8. Although a narrow vertical band is visible around $x \approx 0.5$, this is faint in compar-

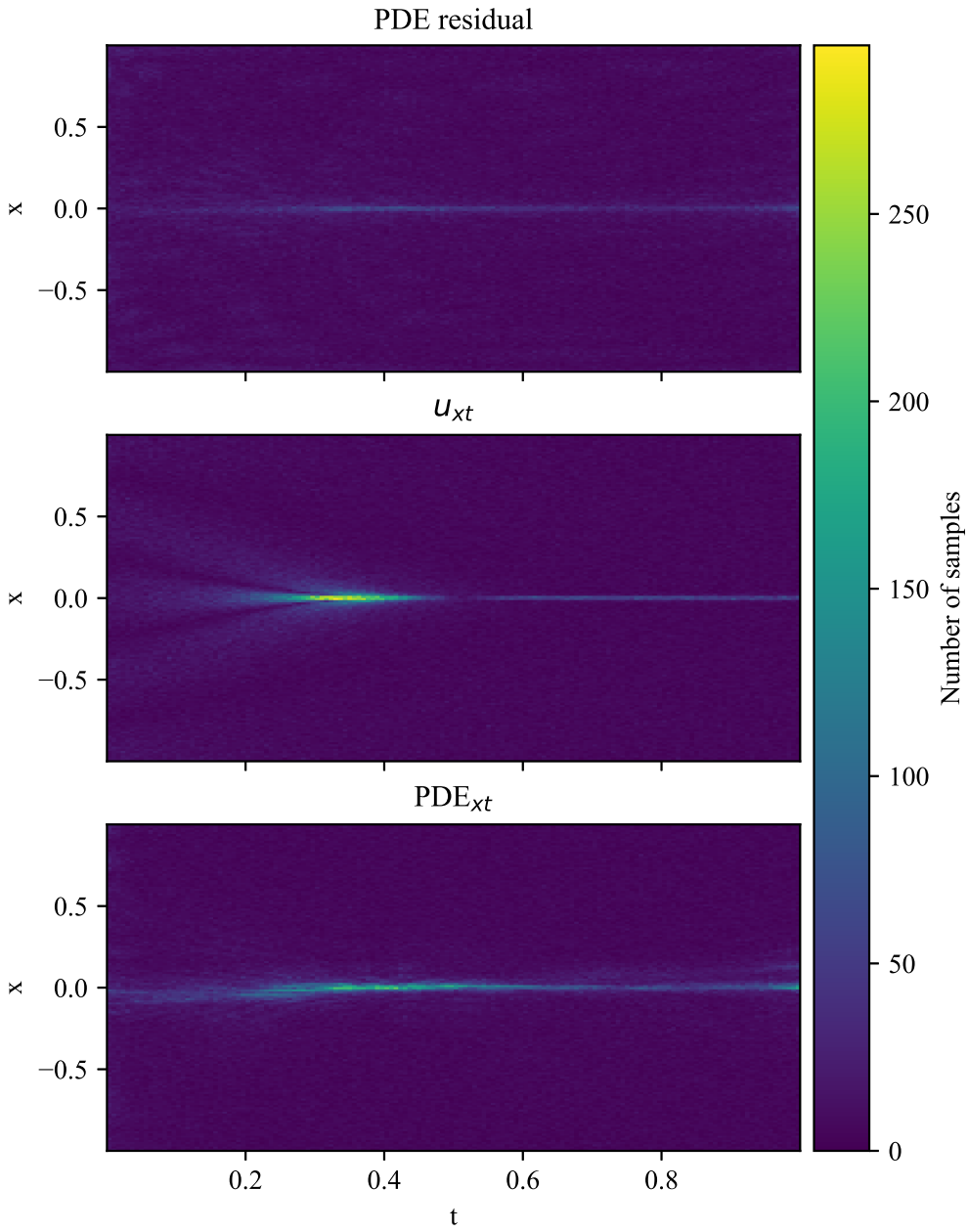

Figure A.1: Two-dimensional histogram showing the cumulative distribution of collocation points selected throughout training according to three distinct guiding metrics for adaptive resampling, as described in Chapter 4. The domain is divided into 150×150 bins, and colour indicates the number of samples that appeared in each spatio-temporal bin. These examples were obtained by solving 1D Burgers' equation using a neural network of depth 3 and 100 resampling iterations, selecting hyperparameters $c = 1$ and $k = 1$.

ison with the upper corners of the domain, which are more heavily resampled at earlier values t, despite the error of both regions being similarly high. The point distribution is more uniform at later stages of t, suggesting that residual-based resampling is primarily driven to the sharper features at early t, with only a few points identifying the vertical band and upper corners at late t.

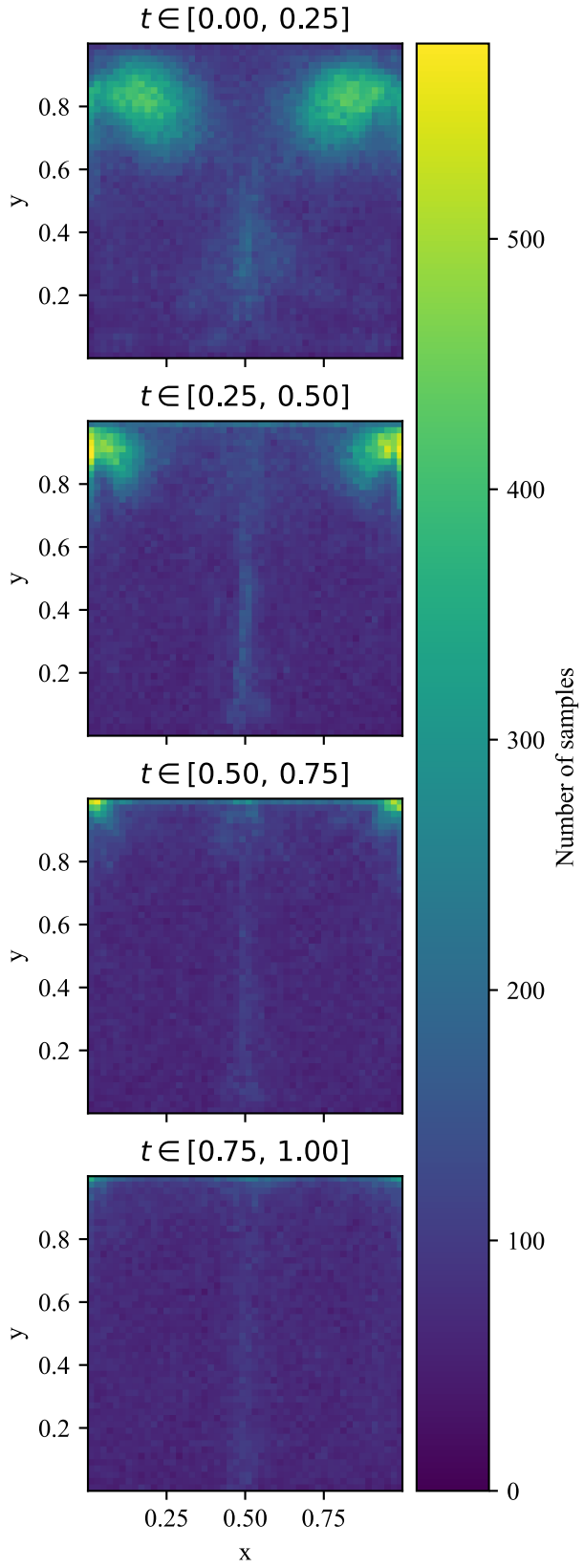

Figure A.2: Two-dimensional histograms showing the cumulative distribution of collocation points selected throughout training according to residual-based resampling, accompanying the model shown in Figure 5.7 and Figure 5.8.

Appendix B

Poisson Equation

B.1 Introduction

The focus of this thesis has been entirely on time-dependent hyperbolic and parabolic problems, and so whilst a variety of different initial conditions and problem parameters have been considered as the test cases for our adaptive collocation point resampling methods, in this appendix we consider the performance of PINNs for an elliptic problem in a non-convex domain.

We consider a relatively simple two-dimensional problem, the Poisson equation, with a constant forcing term of 1 and homogeneous Dirichlet boundary conditions, as defined in Equation (B.1). The problem is made more challenging by the inclusion of an interior square obstacle, resulting in four re-entrant corners at which the gradient of the solution, ∇u, may exhibit singular behaviour. These geometric singularities may provide an additional challenge for PINN-based approximations.

$$\begin{aligned} -\nabla^2 u &= 1 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \Gamma = \partial\Omega, \end{aligned} \tag{B.1}$$

with

$$\Omega = [0,1] \times [0,1] \setminus [0.25, 0.75] \times [0.25, 0.75].$$

As the primary objective remains evaluating methods for the adaptive resampling of interior collocation points, the PINN formulation employs an output transformation to satisfy the boundary conditions exactly, so that the PDE loss to minimise is purely dependent upon the residuals at the collocation points placed in the domain interior. The interior obstacle makes the definition of the output transformation non-trivial, as the simple transformations previously used would also drive the solution to 0 inside the domain. A more sophisticated method following the approach of

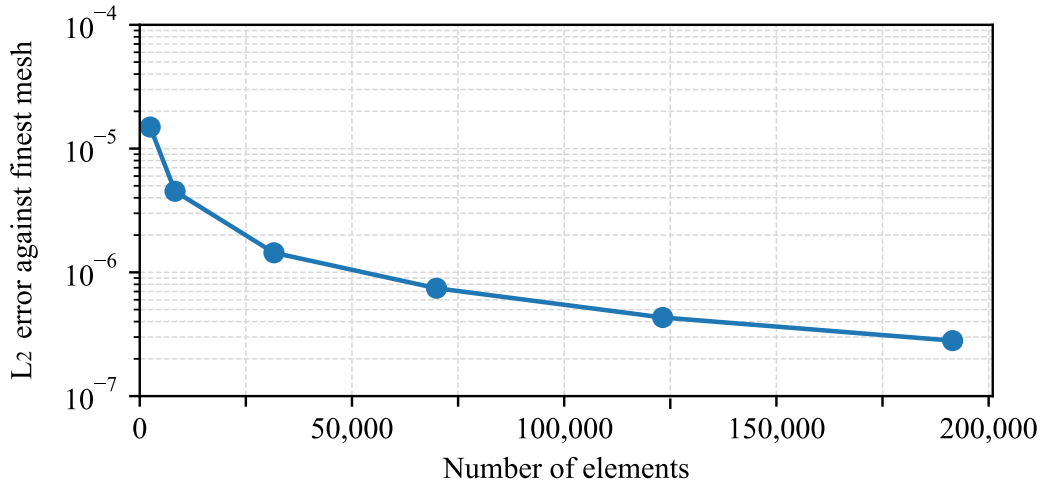

Figure B.1: Mesh independence study of ground truth solution, comparing against finest mesh with 3×10^6 degrees of freedom.

Sukumar and Srivastava [2022] is used, creating an output transformation derived from an approximate distance function (ADF), which both satisfies the boundary conditions exactly on the domain boundary and remains positive and continuously varying everywhere inside the domain.

B.2 Method

To generate a reference solution, the Poisson problem defined in Equation (B.1) was solved using the finite element package Firedrake (Ham et al. [2023]). The computational domain was generated in Gmsh as a structured quadrilateral mesh and imported into Firedrake. The weak form of the governing equation was discretised using continuous quadratic Lagrange finite elements (CG2). Homogeneous Dirichlet boundary conditions were imposed strongly on all domain boundaries and the resulting linear system was solved using Firedrake's PETSc-based solver. Mesh independence (Figure B.1) was verified through successive mesh refinement by comparing solutions against the finest mesh considered, consisting of 500×500 quadrilateral elements and approximately 3×10^6 Degrees of Freedom (DoF).

For the PINN, a neural network of depth 5 and width 64 was used, with tanh activation functions and Glorot-normal weight initialisation. The main deviation from the methods used elsewhere in this thesis arises from the requirement for a more complex output transformation to satisfy the boundary conditions in a hard manner to account for the nested square geometry of the domain. This produced a final transformation shown in Figure B.2 which is sufficient for our requirements, ensuring the solution is strictly positive, continuous, and transforms exactly to 0 at

both interior and exterior boundaries. The transformed solution is obtained simply by multiplying the neural network output by the transformation function,

$$u(x) = \phi(x)N_\theta(x),$$

where $\phi(x)$ denotes the ADF shown in Figure B.2 and $N_\theta(x)$ is the neural network output.

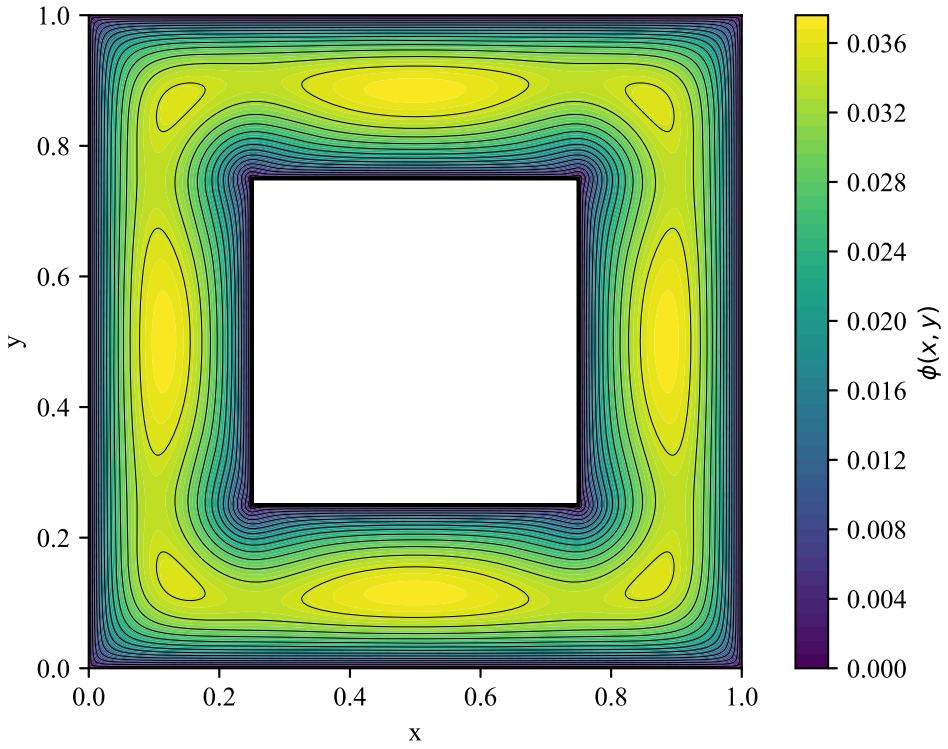

Figure B.2: Contour plot showing the output transformation. The transformation remains continuous and positive throughout the domain and smoothly approaches zero on both interior and exterior boundaries.

Fixed-distribution methods were first considered, with both random and Hammersley point distributions employed as baselines, using 25,000 Adam steps, followed by 50,000 steps of L-BFGS. For adaptive resampling, we used the PDE residual, second mixed derivative u_{xy}, and the mixed derivative of the residual (labelled PDE_{xy}) as adaptive indicators. For adaptive resample, two approaches were implemented: the first carrying out 50 resamples, with 1,000 steps of optimisation with Adam followed by 1,000 steps of L-BFGS for every set of collocation points; and the second approach carrying out 10 resamples, with 5,000 steps of Adam followed by 5,000 steps of L-BFGS. For all methods, we investigated collocation point counts ranging from 1,000 to 10,000, evaluating the mean L_2 error against the reference solution measured over five repeats. Additionally, pointwise error contours are shown to pro-

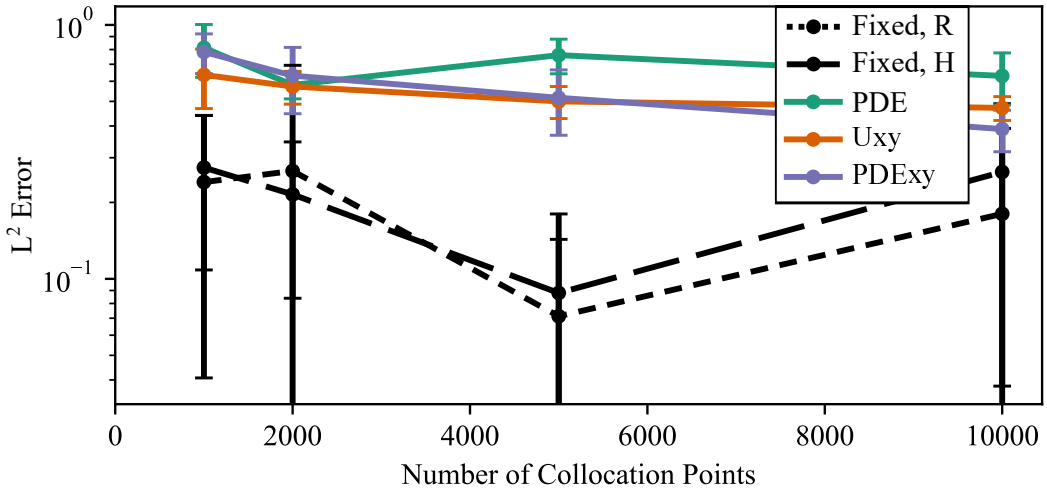

Figure B.3: Error against number of collocation points, using fixed distribution methods and adaptive resampling approaches with a training strategy of 50 resamples and 2,000 total optimisation steps per resample.

vide additional qualitative assessment of individual runs, as well as collocation point density distribution histograms for the different guiding indicators.

B.3 Results

We first examine the overall error achieved as the number of collocation points is varied. The fixed and default adaptive resampling strategy (using 50 resamples) results are shown first in Figure B.3, with Figure B.4 focusing on the adaptive resampling methods. Across all methods, the errors remain relatively high, with no PINN implementation achieving a mean error below 1%. The fixed distribution approaches generally outperform both resampling training strategies, although considerable variance is observed between results. Additionally, modifying the adaptive training strategy from 50 short resamples to 10 longer resamples does not lead to a noticeable reduction in final error. Given the high errors, it is difficult to draw strong conclusions from these quantitative results alone. We therefore next present some qualitative plots obtained from individual simulations, showing solution plots and point distribution histograms.

To further investigate these results, Figures B.5 and B.6 show the predicted solutions and corresponding pointwise errors for the various sampling strategies at $N = 5{,}000$. Consistent with the quantitative results, it is clear that only the fixed approaches generally capture the shape of the solution, and we have to examine the pointwise error to see the location of high error areas. As expected, the error is concentrated near the corners of the interior wall, and is generally higher in the

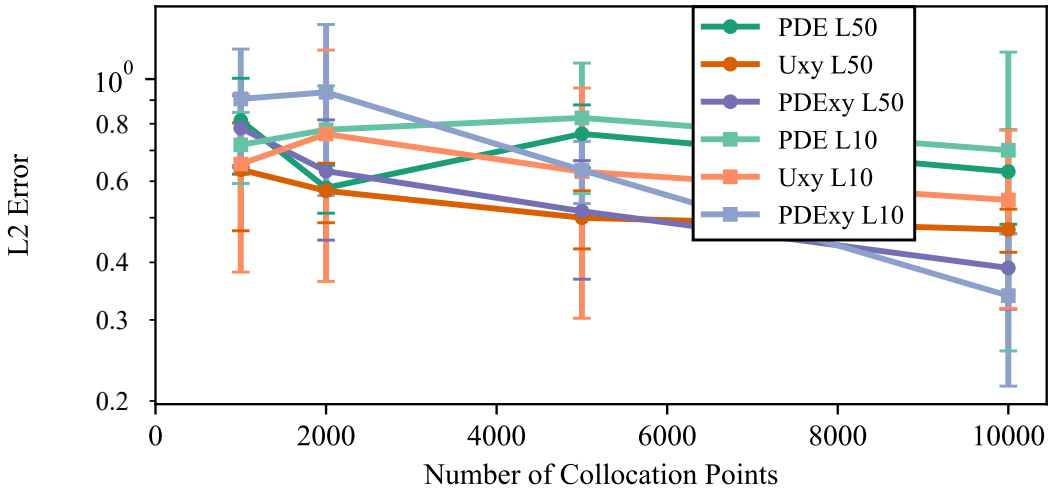

Figure B.4: Error against number of collocation points using two distinct training strategies: 50 resamples with 2,000 total steps per resample (L50), and 10 resamples with 10,000 total steps per resample (L10).

corners of the domain. In the case of the uniform random initialisation, the solution is fairly symmetrical despite us not imposing symmetry in any way. In contrast, the adaptive approaches produce noticeable distortions with no particular symmetry or pattern in the solution. This is also the case for the predictions obtained using the second training strategy (Figure B.6).

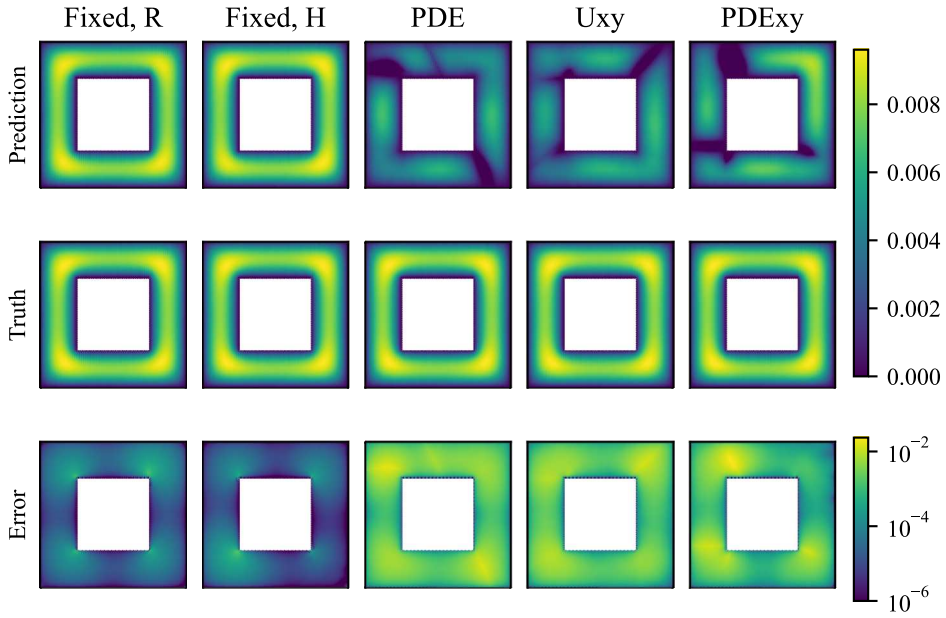

Figure B.5: Predicted solutions compared to reference solution and corresponding pointwise errors for the fixed distribution and adaptive resampling methods using the 50 resample training strategy. Using $N = 5{,}000$ collocation points.

To better understand the behaviour of the adaptive methods, the collocation point density distributions are visualised in Figure B.7. In all cases, the guiding indicators place an increased concentration of points around the corners of the domain,

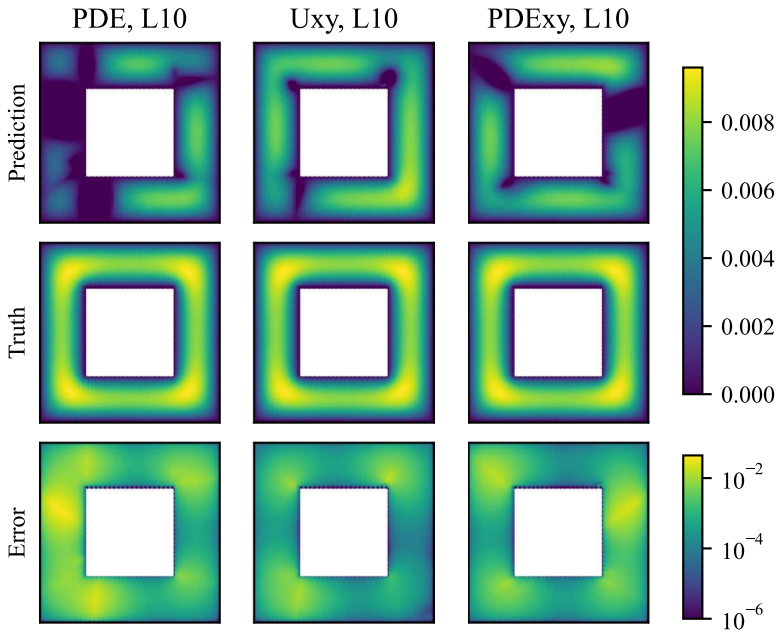

Figure B.6: Predicted solutions compared to reference solution and corresponding pointwise errors for adaptive resampling method using the 10 resample training strategy. Using $N = 5{,}000$ collocation points.

with this behaviour most pronounced for the U_{xy} and PDE_{xy} criteria. Interestingly, despite the very poor solution approximations the U_{xy} indicator is still able to locate and place points near the interior corners. However, no corresponding improvement in solution accuracy is obtained.

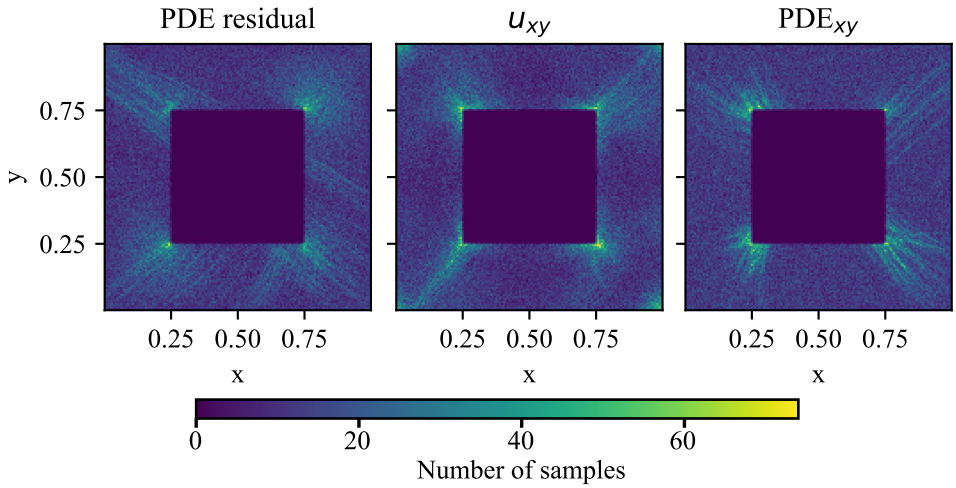

Figure B.7: Two-dimensional histogram showing the cumulative distribution of collocation points selected throughout training using different guiding indicators, obtained from the $N = 5{,}000$, $L = 50$ cases.

To investigate this further, the training and test loss histories are shown in Figures B.8 and B.9 for two examples from the fixed distribution method, and from using an adaptive residual-based resampling method, respectively. Unlike the

examples considered for other PDEs, the test loss never falls below the training loss, suggesting that overfitting occurs across all methods. Such behaviour would be expected for fixed distributions with low N, where not enough points are available to adequately cover the domain, but is observed here across all methods. As shown in Figure B.8, both methods require a large number of optimisation steps before reaching a plateau, indicating a non-trivial optimisation problem. In the $L = 50$ case in Figure B.9, we see that the effect of this on the training loss is that it is unable to be minimised before the training set is changed, effectively making training impossible for the network regardless of the training set chosen.

The modified adaptive strategy partially addresses this optimisation issue. By allowing significantly more optimisation steps between resampling events, the training loss within each resample reduces to approximately 2×10^{-4}, compared to around 10^{-1} in the original approach. However, despite this improved optimisation of the individual training sets, the overfitting issue remains, with test losses remaining high for the $N = 5,000$ example shown here and for other values of N. This suggests that the primary difficulty is not simply insufficient optimisation between resamples, but rather that the underlying PINN implementation is not well equipped to solve this PDE.

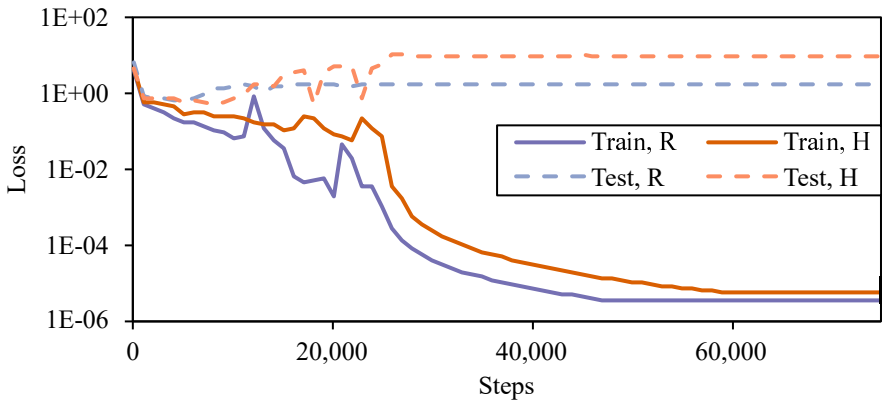

Figure B.8: Loss history obtained using fixed distribution sampling methods with uniform random (R) and Hammersley sequence (H) point distributions.

B.4 Discussion

For this Poisson problem, the simplest approach of using a fixed distribution of collocation points was ultimately the most successful. Although fixed methods remained susceptible to overfitting, avoiding repeated changes to the training set allowed the optimiser to make continuous progress on a consistent objective. As a result, the model is able to learn the broad structure of the solution and, in some cases, pro-

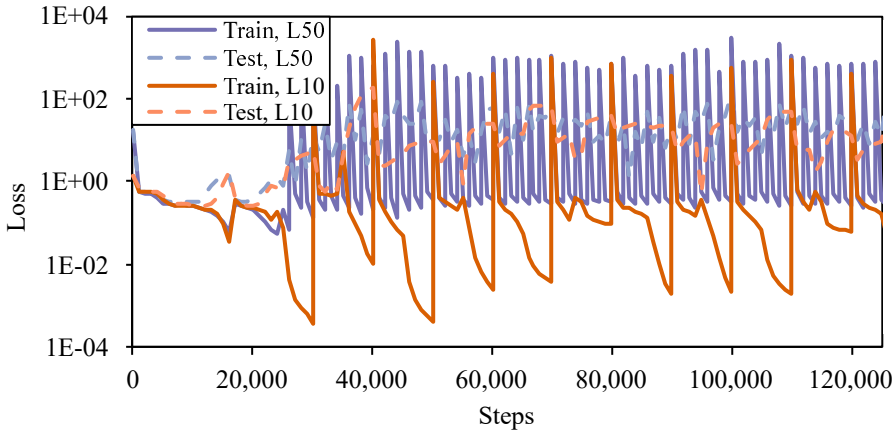

Figure B.9: Loss history obtained using adaptive residual resampling methods with 50- and 10-resample training strategies.

duce qualitatively reasonable approximations, although the low quantitative errors achieved across the other benchmarks studied in this thesis are not achieved.

In contrast, while adaptive resampling methods correctly compute guiding indicators and place points in regions of physical interest, this does not translate into an improvement in solution accuracy. Examination of the loss histories shows that the model is unable to minimise the training loss associated with the current collocation point set before the set is replaced. Consequently, the optimisation objective is repeatedly modified before optimisation for the current set of points has converged, preventing effective progress towards the overall solution.

These observations suggest that, for this Poisson problem, a significant limitation of the adaptive methods lies in the training procedure itself rather than in the identification of physically important regions of the domain. This echoes several of the areas for future work discussed in Chapter 7.2.2, particularly the need for approaches capable of automatically adapting training strategies to problem difficulty and maintaining effective optimisation as problem complexity increases. For the present problem, a resampling strategy triggered only after the loss has plateaued may prove more effective (as in Subramanian et al. [2023]), as it would reduce the likelihood of optimisation being interrupted before sufficient progress has been made on the current collocation set.

A secondary consideration specific to this problem concerns the ADF-derived method employed to enforce the boundary conditions exactly. As discussed in Section 10.1 of Sukumar and Srivastava [2022], the Laplacian of the ADF becomes unbounded near the vertices of internal polygon corners. In practice, this restricts the placement of collocation points close to portions of the boundary when collocation-based methods involving second derivatives are employed. Although it is likely that

this represents a comparatively minimal source of error in the present investigation, a more comprehensive study would likely need to address this issue in order to fully exploit the adaptive point placement near the domain boundaries and improve the approximation of the solution in the vicinity of the interior corners.

Additionally, an important observation concerns the nature of the PDE itself. Compared to the other principal benchmark considered in this thesis, Burgers' equation, the Poisson problem may be inherently more challenging for standard PINN architectures. Tanh activation functions, which we have used, are commonly employed in PINNs due to their smooth differentiability. Since the Burgers' equation solution profiles can be expressed in terms of exponential and hyperbolic tangent functions, the inductive bias of tanh-activation funtions may naturally align with the underlying solution. As no such advantage is present for the Poisson problem considered here, it may be the case that employing a similarly sized network architecture and number of optimisation steps may be insufficient.

While the benchmarks examined elsewhere in the thesis demonstrate that adaptive resampling can improve accuracy and reduce the number of collocation points required to achieve a given error level, the results in this appendix indicate that such benefits rely on the underlying model having adequate approximation capability. And whilst adaptive sampling may improve the quality of the training set, care must be taken in its implementation to ensure that adaptive resampling enhances rather than impedes the optimisation procedure.